NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

NPSNET: HIERARCHICAL DATA STRUCTURES FOR REAL-TIME THREE-DIMENSIONAL VISUAL SIMULATION

by

Randall Lee Mackey

September 1991

Thesis Co-Advisors: Michael J. Zyda
David R. Pratt

Approved for public release; distribution is unlimited.

RITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
DECLASSIFICATION/DOWNGRADING SCHEDULE	
ERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION mputer Science Dept. val Postgraduate School — 6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
ADDRESS (City, State, and ZIP Code) onterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

TITLE (Include Security Classification) NPSNET: HIERARCHICAL DATA STRUCTURES FOR REAL-TIME THREE-DIMENSIONAL VISUAL SIMULATION (U)

PERSONAL AUTHOR(S) Mackey, Randall Lee

. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 11/90 TO 09/91	September 1991	91

SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the icial policy or position of the Department of Defense or the United States Government.

COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
IELD	GROUP	SUB-GROUP	Computer Graphics, Simulators, Hierarchical Data Structures, Digital Terrain Data

ABSTRACT (Continue on reverse if necessary and identify by block number)

NPSNET is a low-cost visual simulation system designed and constructed at the Naval Postgraduate School. PSNET uses digital terrain data and renders scenes involving vehicles, aircraft, cultural features, and natural tures in real-time. The implementation of a terrain paging algorithm in NPSNET is discussed. Terrain paging pands the terrain area available for simulation and overcomes the limits of main memory size. Hierarchical data uctures commonly used in visual simulation systems are surveyed. The generation of a multi-resolution terrain taset and the implementation of a hierarchical data structure are explained. The multi-resolution dataset is created generating lower resolution descriptions of polygons from the original data. The hierarchical data structure used NPSNET, based on quadtrees, provides a means to attenuate the resolution of terrain over distance and cull those rtions of terrain outside of the user's field of view.

DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
. NAME OF RESPONSIBLE INDIVIDUAL chael J. Zyda	22b. TELEPHONE (Include Area Code) (408) 646-2305 — 22c. OFFICE SYMBOL CS/Zk

FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED

Approved for public release; distribution is unlimited

NPSNET: HIERARCHICAL DATA STRUCTURES FOR REAL-TIME THREE-DIMENSIONAL VISUAL SIMULATION

by
Randall Lee Mackey
Captain, United States Army
B.S., United States Military Academy, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

ABSTRACT

NPSNET is a low-cost visual simulation system designed and constructed at the Naval Postgraduate School. NPSNET uses digital terrain data and renders scenes involving vehicles, aircraft, cultural features, and natural features in real-time. The implementation of a terrain paging algorithm in NPSNET is discussed. Terrain paging expands the terrain area available for simulation and overcomes the limits of main memory size. Hierarchical data structures commonly used in visual simulation systems are surveyed. The generation of a multi-resolution terrain dataset and the implementation of a hierarchical data structure are explained. The multi-resolution dataset is created by generating lower resolution descriptions of polygons from the original data. The hierarchical data structure used in NPSNET, based on quadtrees, provides a means to attenuate the resolution of terrain over distance and cull those portions of terrain outside of the user's field of view.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. MOTIVATION

Visual simulation systems in the form of vehicle, flight, and battlefield simulators continue to become more widespread in the military. Simulators provide training at lower cost and less risk than by other means. As technology improves, the quality, realism, and effectiveness of visual simulation systems also improves.

One simulation system designed by the Defense Advanced Research Projects Agency (DARPA) is Simulation Networking (SIMNET) (Thorpe, 1987). SIMNET is a vehicle and battlefield simulation system designed to train vehicle crews for combat. SIMNET nodes consist of visual displays and a mock-up of the crew compartment of a particular vehicle. An effort to develop a low-cost system that will be able to interface with SIMNET nodes over a network has been ongoing at the Naval Postgraduate School (NPS). This system, NPSNET (Zyda and Pratt, 1991), is currently in a developmental stage and consists of Silicon Graphics IRIS Workstations that communicate over an Ethernet local area network.

B. FOCUS

Many factors determine the quality of a real-time 3D visual simulator. Two of these are the fidelity of the actual scenes rendered and the frame rate of the simulator. High fidelity scenes provide a greater degree of realism and enhance the effectiveness of a simulator as a training device or visualization tool. High frame rates, in excess of 15 frame per second, make motion in a simulator smooth and also improve interaction with the simulator. These two factors are at odds with each other; rendering more detailed scenes

takes more time and hence slows the frame rate. Attempts to increase the frame rate allow less time to render detailed scenes in individual frames. An approach to building a real-time visual simulator that attempts to balance these two factors is to add as many features that enhance realism, while still keeping the frame rate above the threshold at which the human eye is able to detect individual frames.

Speed and efficiency are the primary considerations in producing code for a visual simulator. Designers must choose data structures and algorithms that support efficient code. Specifically this work focuses on two main goals pertaining to NPSNET. The first goal is to expand the terrain area used in the simulation and the second is to reduce polygon flow while rendering terrain scenes.

1. Expanding the Terrain Area for Simulation

The first goal involves allowing the simulator to use all of the terrain data available and not limit the simulation to some small area, even though data for a larger area is available. Since memory size is limited, having all of the data in main memory is not feasible; therefore, paging terrain data through some sort of dynamic algorithm is required. Early developmental versions of NPSNET limited the simulation to an eight by eight kilometer area, even though data was available for a 50 by 50 kilometer area. Previous simulators developed at the Naval Postgraduate School do not implement dynamic terrain paging or force the user to switch from a three dimensional display to a two dimensional display in order to move to a new active area.

2. Reducing Polygon Flow

The second goal concerns the implementation of a hierarchical data structure that supports rapid culling of polygons not within the field of view and provides multiple resolutions of terrain. Both of these benefits assist in reducing the number of polygons composing each individual frame. The relation between the number of polygons in an individual frame and the frame rate is simple--the fewer the number of polygons the faster

the frame rate. Representing terrain and objects farther from the viewer with fewer polygons (lower resolution) reduces the total number of polygons in the scene. Figure 1.1 (Jurewicz, 1990, p. 14) shows the relationship between distance, the number of polygons, and multiple resolution. This type of representation provides another benefit; since terrain and objects at a distance are represented with fewer polygons, it is more difficult to discern the true form of these items. This models the effect of viewing distant objects in the real world.

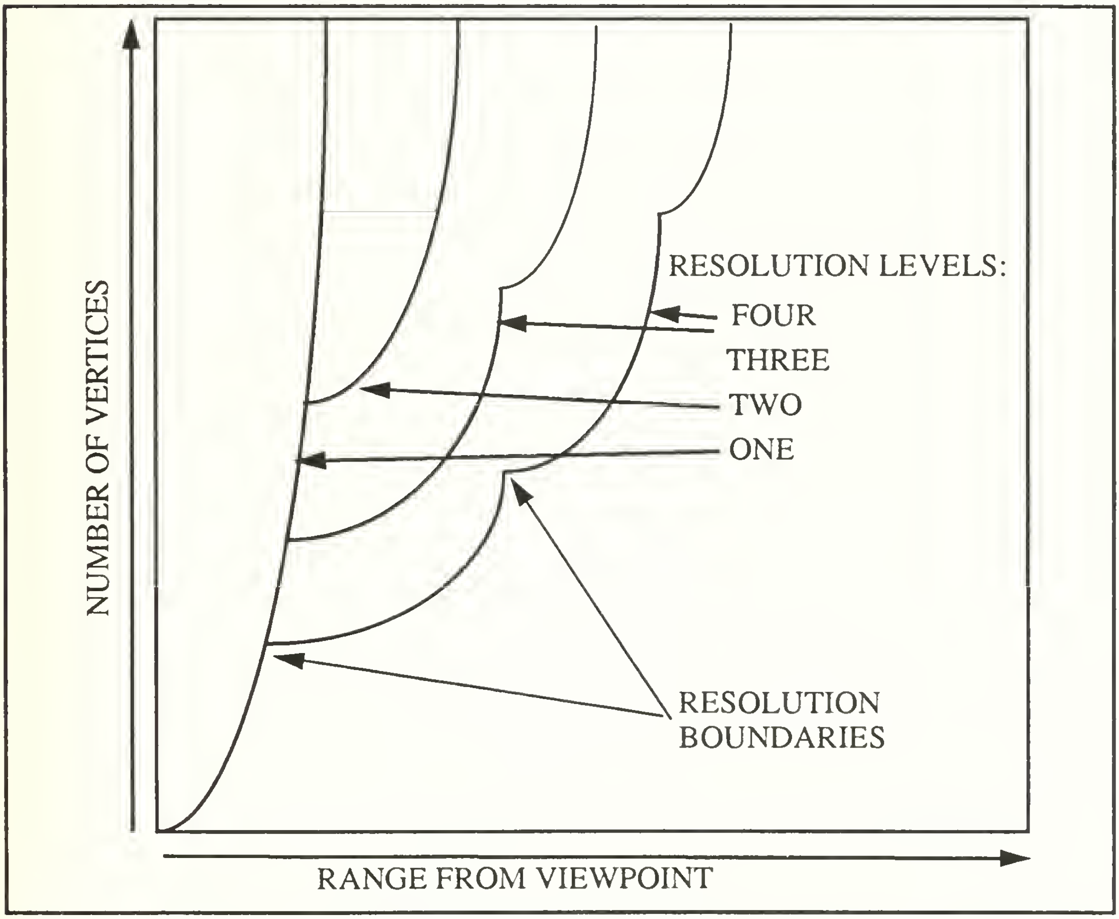


Figure 1.1. Relationship Between Number of Polygons, Distance, and Multiple Resolution

C. SUMMARY OF CHAPTERS

Chapter II of this thesis examines previous simulators developed at the Naval Postgraduate School and focuses on data structures and implementations of multiple resolution. Attempts to expand the simulation area to the entire set of data available are discussed in the systems to which this pertains. Chapter III is an overview of NPSNET and provides background on the capabilities of the system. Chapter IV discusses the implementation of terrain paging in NPSNET.

Chapter V is an overview of hierarchical data structures commonly used in computer graphics. These include quadtrees, octrees, and binary space partitioning trees. Chapter VI focuses on the implementation of the hierarchical data structure chosen for NPSNET--one based on quadtrees. Chapter VII discusses terrain rendering in NPSNET, with emphasis on how the hierarchical data structure supports efficient scene rendering. Chapter VIII is a summary of results and conclusions, and provides suggestions for future work in these areas.

Two appendices are provided. Appendix A details the attributes which describe terrain in the dataset used by NPSNET. Appendix B explains the format of the terrain data files.

II. HISTORY OF REAL-TIME VISUAL SIMULATORS AT NPS

Real-time interactive 3D graphics is the focus of the Graphics and Video Laboratory at the Naval Postgraduate School. Over the last five years, students and faculty have produced a series of real-time visual simulators. Simulators improved in quality as the graphics workstations available increased in capabilities and processing power. In order to understand the approaches taken in NPSNET, these previous simulators need to be discussed. In looking at these systems the focus is on how terrain paging, multiple resolution, and polygon culling were accomplished.

A. FOGM MISSILE SIMULATOR

1. Overview

The Fiber Optically Guided Missile (FOGM) simulator (Smith and Streyle, 1987) was the first major simulator effort at the Naval Postgraduate School. The simulator allows the user to see a three dimensional view from a user-controlled missile as it flies over a fixed ten by ten kilometer area. The user is able to target, track, and destroy vehicles present in the simulator. The user selects a ten by ten kilometer active area from a larger 35 by 35 kilometer area when the simulation begins. The simulator uses a special elevation dataset for Ft. Hunter Liggett, California. The data points for this dataset are every 12.5 meters apart, while normal Defense Mapping Agency Digital Terrain Elevation Data (DMA DTED) Level 1 datasets only contain data points every 3 arc seconds (approximately every 100 meters) (Defense Mapping Agency, 1986, p. 1). Use of all the data points in the dataset is not feasible due to the number of polygons created, so only every sixth data point is used. This effectively makes the resolution of the data for the

simulator one point every 75 meters. The dataset also contains vegetation height data at each data point, but this data is not used. The scenes in the simulator contain no natural objects or cultural features. The simulator does contain other vehicles, but these vehicles are not under the user's control. The FOGM simulator was initially designed for a Silicon Graphics, Inc. IRIS 3120 workstation. This workstation does not have hardware support for simultaneous use of double buffering and z-buffering (for hidden surface elimination), so the Painter's Algorithm is used for hidden surface removal.

The coordinates of each vertex of the polygons composing the terrain skin are stored explicitly in an array. When the user selects a ten by ten kilometer active area, explicit polygon descriptions are built from the dataset. This causes some elevation points to be replicated up to six times, once for each of the six triangles that share a vertex as shown in Figure 2.1. The color of the terrain for each 75 meter square is also stored in a corresponding array.

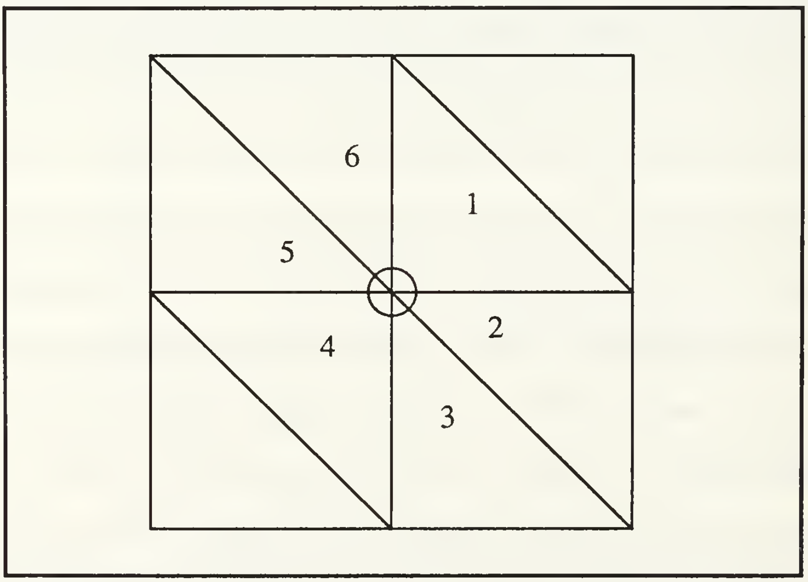


Figure 2.1. Six Triangles Sharing a Common Vertex

2. Terrain Paging

The FOGM simulator does not use any type of terrain paging and the user is restricted to the ten by ten kilometer area chosen even though data is available for a larger 35 by 35 kilometer area. The user can explicitly change the active area of the simulator by exiting to a high level menu and choosing a new ten by ten kilometer active area.

3. Polygon Culling

The FOGM simulator performs polygon culling by selecting the rectangular area within the active area of the simulator that encompasses the field of view. This means that some polygons not within the field of view are still rendered as shown in Figure 2.2.

B. VEH VEHICLE SIMULATOR

1. Overview

The VEH vehicle simulator (Oliver and Stahl, 1987) was developed from the FOGM simulator and contains many of the same features. The VEH simulator allows the user to maneuver a vehicle over terrain in real-time. The dataset and selection of a ten by ten kilometer active area are the same as in the FOGM simulator. Like the FOGM simulator, the coordinates of the vertices of each polygon composing the terrain skin are stored explicitly in an array. A polygonal description of the terrain is built from the dataset just as in the FOGM simulator.

2. Terrain Paging

Like the FOGM simulator, the VEH simulator makes no use of terrain paging. The user is confined to the ten by ten kilometer active area selected from the larger 35 by 35 kilometer dataset. Changing the active area of the simulator has to be done explicitly.

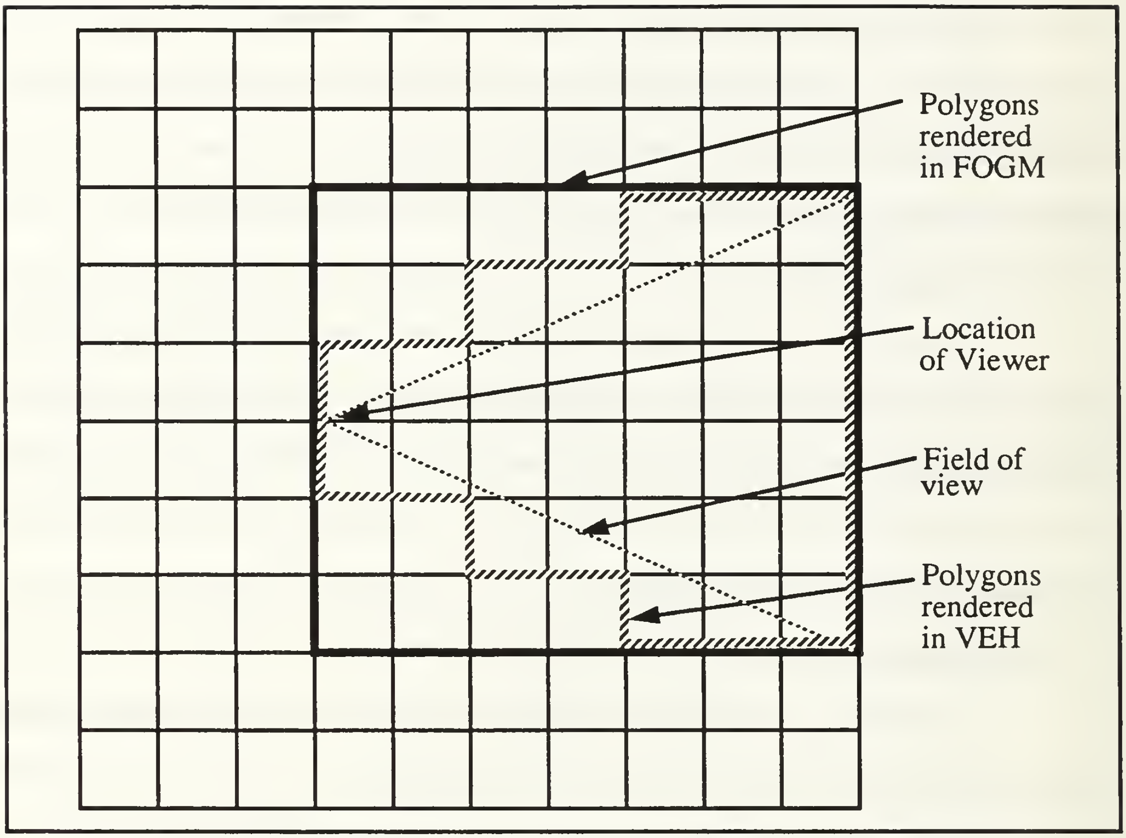


Figure 2.2. Field of View Versus Polygons Rendered in FOGM and VEH Simulators

3. Polygon culling

The culling of polygons outside the field of view is improved in VEH. The VEH simulator culls polygons outside the field of view in order to limit the number of polygons rendered in each frame. Polygons are only rendered if they pass a check against the slope of the lines delineating the left and right limits of the field of view. The 360 degree field of view is divided into octants, so there are eight different cases which determine how the array containing the terrain data is processed. This is an effective means for polygon culling; however, all terrain within the field of view from the user's location to the edge of

the active area is rendered. Figure 2.2 shows a comparison of the terrain rendered in the FOGM simulator and in VEH.

C. MPS MOVING PLATFORM SIMULATORS

1. Overview

The MPS Moving Platform Simulators were developed over three generations: MPS (Fichten and Jennings 1988), MPS II (Winn and Strong, 1989), and MPS III (Cheeseman, 1990). The second and third generations of this simulator are modifications and improvements of the original MPS simulator . The MPS simulator allows the user to drive or fly platforms over three dimensional terrain. The simulator allows the user to select a ten by ten kilometer active area from a larger dataset. The simulator can use one of two datasets: a standard DMA DTED Level 1 datafile describing approximately a 120 by 120 kilometer area or the special higher resolution dataset used by the FOGM and VEH simulators that describes a 35 by 35 kilometer area. MPS allows the user to vary lighting parameters to simulate the changes in sunlight throughout the day as well as seasonal changes due to the relative positions of the earth and sun.

The MPS II simulator is the next generation in this family of simulators. MPS II uses only the special high resolution terrain dataset and can not access standard DMA DTED Level 1 data. The MPS II simulator implements a method to vary the intervals for the multiple resolutions of terrain and can display terrain using data points at 12.5, 25, 50, 75, and 100 meter intervals. MPS II uses mesh drawing primitives to render terrain, rather than explicitly rendering each polygon. MPS II also implements an intervisibility display to determine the visibility over a line-of-sight between two points.

MPS III attempts to combine the best features of MPS and MPS II. MPS III allows the user to access either DMA DTED Level 1 datasets or special high resolution

datasets. This simulator also allows the user to choose whether to render terrain using the mesh primitives or to use polygon rendering primitives.

2. Terrain Paging

Like their predecessors, the MPS simulators do not use any type of terrain paging. Changing the active area of the simulator has to be done explicitly by exiting to a high-level menu and choosing a new ten by ten kilometer active area.

3. Multiple Resolution

MPS is the first simulator developed at the Naval Postgraduate School to use multiple resolution of terrain. MPS displays terrain at three different resolutions. The user can select to render all terrain at the one resolution or select a multiple resolution option. In the multiple resolution mode, 100 meter squares are displayed near the viewer, 200 meter squares are displayed for 2000 meters, and 400 meter squares are displayed to the edge of the active area. These squares are all composed of two triangles. The method used by MPS to determine which areas to draw at each resolution is simple, but not very effective. The algorithm used works well when the direction of view is close to 0, 90, 180, or 270 degrees, but not well at other angles. The algorithm is too closely tied to the rectangular structure of the array and the range of each level of resolution is not independent of the direction of view as show in Figures 2.3 and 2.4.

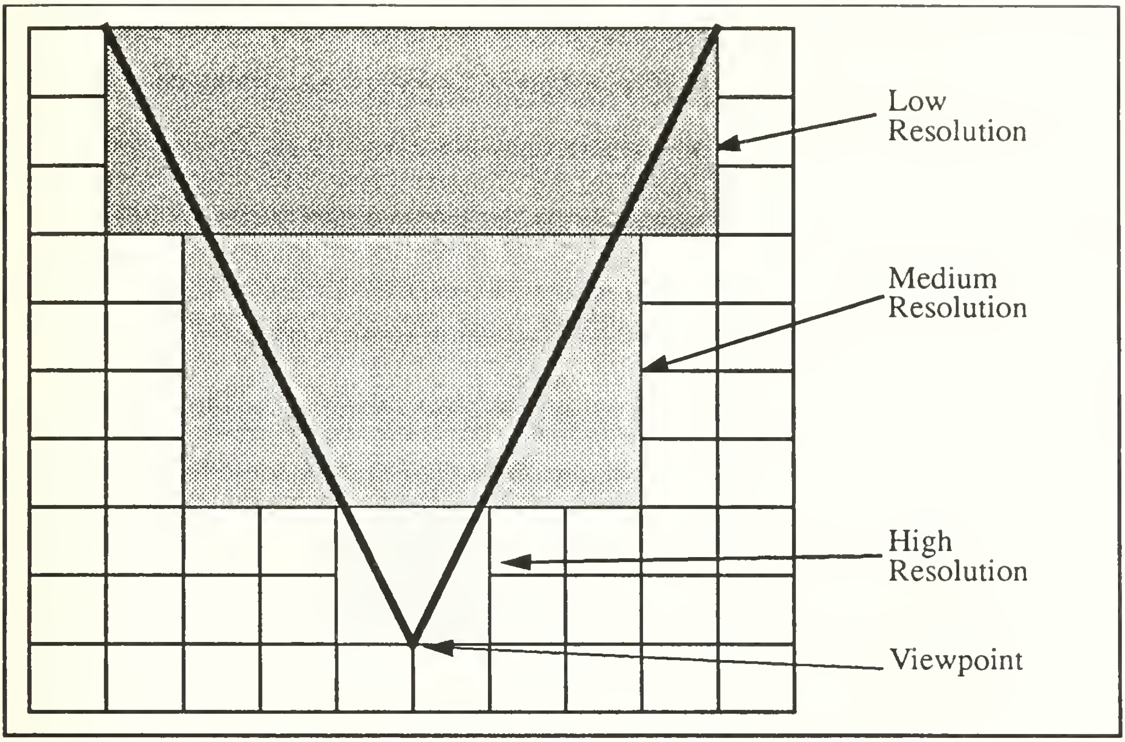


Figure 2.3. Multiple Resolution Regions in MPS

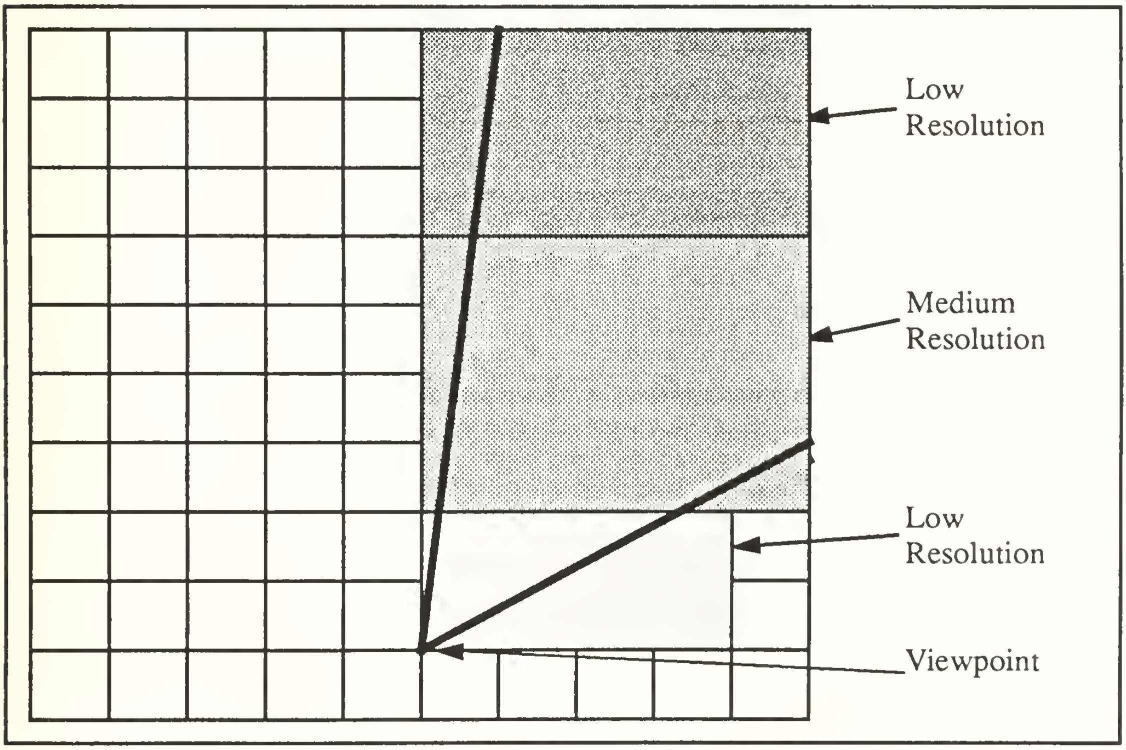


Figure 2.4. Multiple Resolution Regions in MPS with Different Direction of View

When using polygon drawing primitives and multiple resolution in this way, gaps can develop in the terrain skin at the boundaries of polygons of different resolutions as show in Figure 2.5. In MPS, these gaps are filled by additional polygons of the same material color and using the same normal as the adjacent polygon of lower resolution.

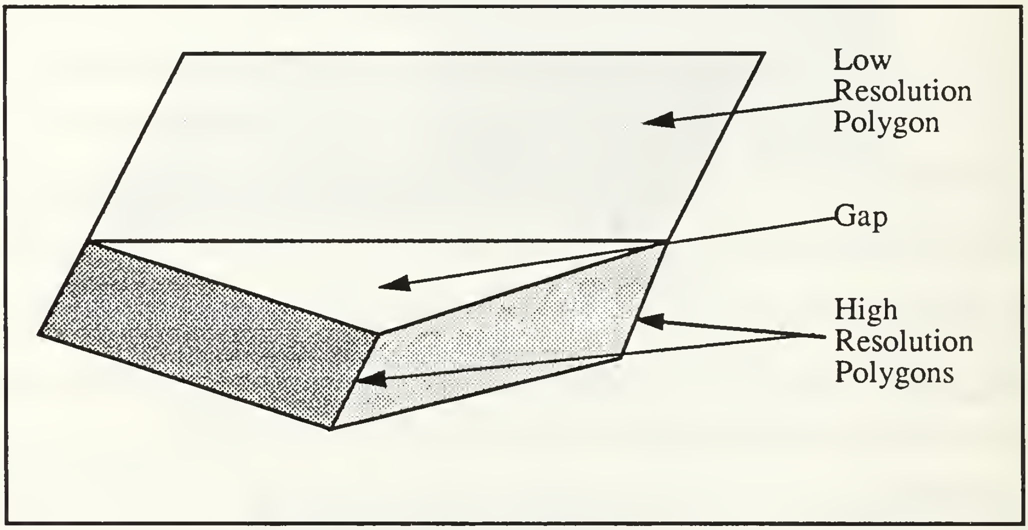


Figure 2.5. Gap Created at Resolution Boundaries

MPS does not use multiple resolutions of vehicles placed on the terrain surface. Additionally, vehicles are always rendered as if they were on terrain rendered at the 100 meter resolution. No adjustments are made for the fact that the characteristics of the underlying terrain can change when terrain is rendered at progressively lower resolutions.

MPS II uses mesh drawing primitives, but approaches multiple resolution using the same algorithms as MPS. Multiple resolution is accomplished using mesh drawing primitives by simply not using every data point. Using every other point provides a lower level of resolution, while using every fourth point provides yet a lower level, and so on. The problem of gaps at the boundaries of different resolutions is handled in the same

way as in MPS, but note this is done in conjunction with terrain rendered using mesh primitives. MPS III simply offers a choice between the methods used by MPS and MPS II.

4. Polygon Culling

The MPS simulator culls polygons outside the field of view in order to limit the number of polygons rendered in each frame. First the point of view is offset in order to ensure enough terrain is rendered near the edge of the field of view. A bounding box is established around the triangle composing the field of view at each resolution as shown in Figures 2.3 and 2.4. Only polygons within this bounding box are rendered. This is an effective means for polygon culling; however, some polygons outside the field of view are still rendered and all terrain between the viewer's location and the edge of the active area is rendered.

MPS II and MPS III use the same culling algorithms as MPS. The algorithm used works while rendering terrain as polygons or as a mesh.

D. CCWF COMMAND AND CONTROL WORKSTATION

1. Overview

The CCWF simulator (Weeks and Phillips, 1989) is a tool designed to assist a tactical commander by providing a three-dimensional view of his immediate area of operation and a two-dimensional view of the underwater terrain directly beneath his vessel. The simulator displays surface and subsurface ships. The simulator is networked, allowing users on different workstations to maneuver different ships simultaneously. The simulator uses standard DMA DTED Level 1 datasets for an area of Japan. The designers assumed a point spacing of 100 yards effectively making each data set describe a 60 by 60 nautical mile area. This assumption holds for data cells near the equator. The user can place vessels anywhere within the 60 by 60 nautical mile area by zooming in to any particular five by

five nautical mile area and placing the vessels as he desires. No natural features or cultural features are placed on the terrain.

CCWF allows the user to choose whether terrain is rendered using polygon or mesh drawing primitives. If the user chooses the polygon drawing primitives, the terrain is drawn as a checkerboard of colors to provide a sense of motion and assist in depth perception.

2. Terrain Paging

CCWF provides a larger active area for simulation than previous simulators, but does not perform true terrain paging. Only a five by five nautical mile area is displayed in a two-dimensional situation map at a given time, but the data for an entire 60 by 60 kilometer area is in main memory. No additional terrain data is paged in when the vehicle maneuvered by the user nears the edge of the active area.

3. Polygon Culling

CCWF effectively culls polygons outside the field of view using a unique method. The designers of CCWF divided the field of view into 72 five degree sectors. The slopes of the edges of the sectors are stored in a lookup table. These values are used to determine which sectors to render. CCWF uses a default of 45 degrees for the field of view and nine of these sectors are rendered in each frame for this field of view.

4. Multiple Resolution

Multiple resolution is implemented in CCWF by not using every data point in the lower resolutions. Every point is used in high resolution, every other point in the next resolution, and every fourth point in the next resolution. The boundaries of the resolutions are fixed within each sector making up the field of view. This is an improvement over the MPS family of simulators which use fixed boundaries within each of the four quadrants.

CCWF creates filler polygons to fill the gaps created at resolution boundaries similar to the method used in the MPS simulators. These fill polygons are generated in real-

time from the data already in memory. Some problems with scene quality in the CCWF simulator are visible when using fill polygons while rendering terrain as a mesh (Jurewicz, 1990, pg. 7).

E. NPS AUTONOMOUS UNDERWATER VEHICLE SIMULATOR

1. Overview

An ongoing, interdepartmental project at the Naval Postgraduate School has been to develop an Autonomous Underwater Vehicle (AUV). The AUV simulator (Jurewicz, 1990) models the prototype vehicle's dynamics and provides a simulator for the AUV in a test environment or in open water. The test environment for the AUV is the Naval Postgraduate School swimming pool and the AUV simulator's performance can be compared against actual data gathered from tests of the AUV. The AUV simulator allows the user to maneuver the AUV in the chosen environment by manipulating virtual control devices on the screen. The control panels were developed using the NPS Panel Designer (King and Prevatt, 1990). The open water terrain data for the simulator was obtained from the Monterey Bay Research Institute and comprises a 22 by 26 nautical mile area of the bay. The simulator renders terrain using mesh primitives and introduces some new approaches to rendering terrain. The terrain can be textured with a digital image, providing a sense of motion and depth perception on terrain rendered as a mesh. This is a significant improvement over earlier simulators that use mesh drawing primitives, and provides a different option than checkerboarded terrain to provide a sense of motion. Texturing produces high quality scenes; however, it does affect system performance.

The AUV simulator does not place any natural or cultural features on the terrain. The designer notes some problems with attempting to place objects on terrain that is not rendered in the highest resolution (Jurewicz, 1990, pg. 27).

2. Terrain Paging

The AUV simulator does not utilize terrain paging. The Monterey Bay dataset consists only of elevation values and the entire dataset is read into an array at the beginning of the simulation. No facilities are provided for paging additional terrain once the edge of the active area is reached.

3. Polygon Culling

The AUV simulator renders all terrain within a bounding box that encompasses the field of view. This means that some polygons not in the field of view are rendered, but the designer notes that the performance of the simulator is better using a simple "rough clip" of the polygons not in the field of view, rather than performing additional computation to discern which polygons are actually visible (Jurewicz, 1990, pg. 13). This method of terrain culling works well in the AUV simulator because terrain is rendered as a mesh.

4. Multiple Resolution

The AUV simulator not only displays terrain at multiple resolutions, but allows the user to vary parameters which determine how many resolutions of terrain are rendered and how far out terrain is rendered. The simulator also has a dynamic mode in which these parameters are automatically adjusted in order to keep the frame rate and the quality of the terrain rendered within acceptable limits. For example, if the frame rate is above a certain threshold the simulator will render more terrain at higher resolutions. If the frame rate falls below a certain threshold, the boundaries of the different resolution levels will be adjusted and more terrain will be rendered at lower resolution levels.

The AUV simulator approaches the problem of boundaries between resolution levels in a new way. No fill polygons are generated at these boundaries; rather, the seams are "stitched" together by selecting certain vertices to send to the mesh drawing routine as shown in Figure 2.6. This provides a smooth transition between resolution levels.

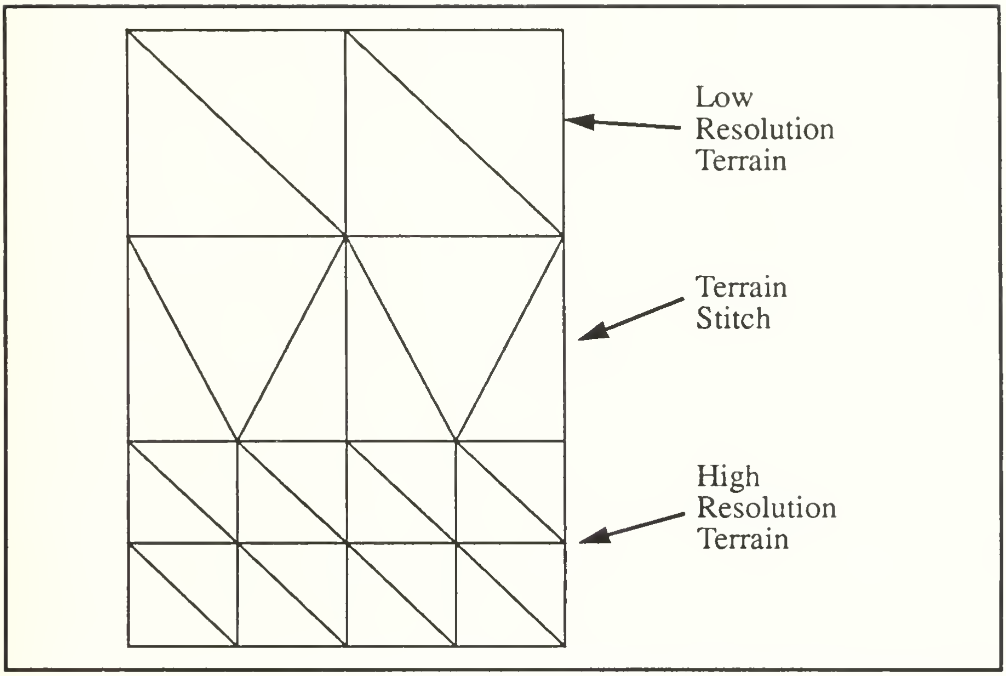


Figure 2.6. Seam Stitching Technique used in AUV Simulator

The boundaries between resolution levels remain rectangular in nature. This results in varying distances from the viewer to particular resolution boundaries depending on the direction of view. This is depicted in Figure 2.7. Figure 2.8 shows an overhead view of the active area of the AUV simulator rendered as a grid and the orientation of the rectangular boundaries of the resolution areas is evident.

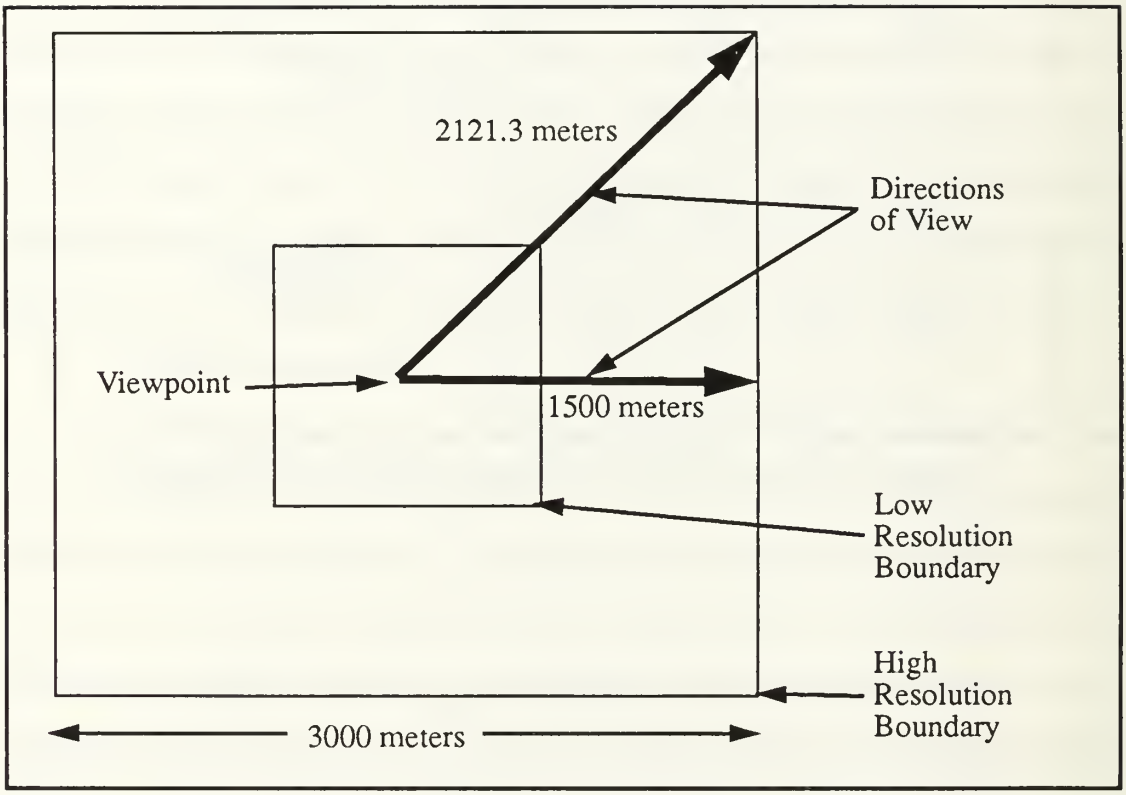


Figure 2.7. Problem with Range of Resolutions Based on Rectangles

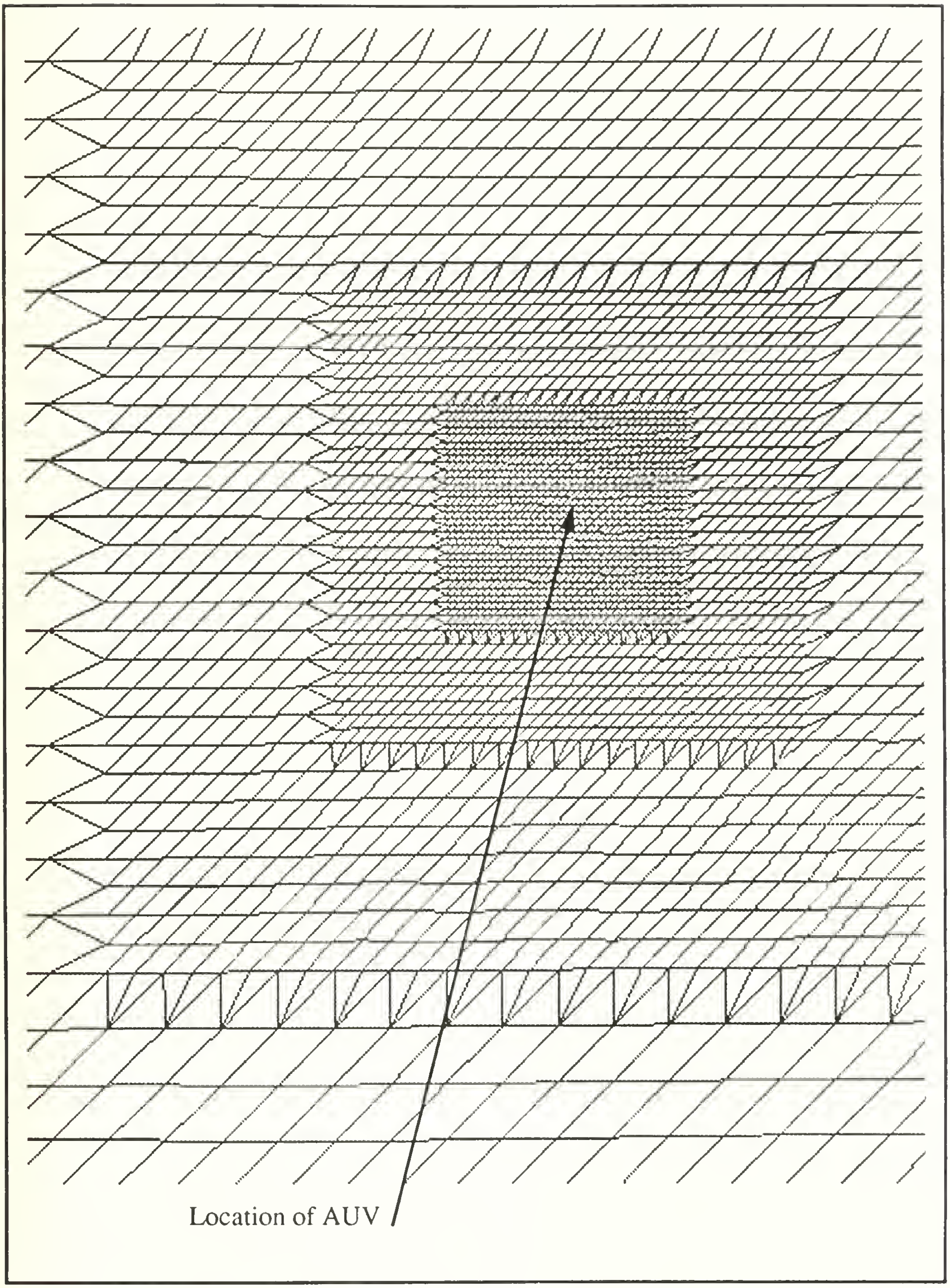


Figure 2.8. Multiple Resolution Areas in AUV Simulator

III. NPSNET

NPSNET is a real-time three-dimensional visual simulator. It is designed as a low-cost node for SIMNET. The intent of NPSNET is to provide such a node using commercially available graphics hardware and to communicate over a network to other SIMNET nodes.

NPSNET allows the user to select and maneuver platforms over terrain. NPSNET currently contains a wide selection of platforms including vehicles, aircraft, and ships. The system is networked, allowing individuals to interact from different workstations. The appropriate scene, relative to the vehicle being maneuvered, is displayed on each workstation. The user can interact within the environment by maneuvering the selected platform, firing munitions at other platforms, or simply exploring the "virtual" Fort Hunter Liggett. Figure 3.1 shows a view from a vehicle in NPSNET.

A. TERRAIN DATASET

The primary concern of this effort is how the terrain dataset is used in NPSNET. The use of actual terrain data in visual simulators is a relatively recent occurrence (Schachter, 1983, p. 138). The dataset used in the developmental version of NPSNET consists of terrain and feature data for a 50 by 50 kilometer area of Fort Hunter Liggett, California. BBN Systems and Technologies provided the dataset. The dataset conforms to the SIMNET Database Interchange Specification (SDIS) (Lang and Wever, 1990). SDIS has provisions for more data than is used in NPSNET; not all features of SDIS are contained in the Fort Hunter Liggett Release.

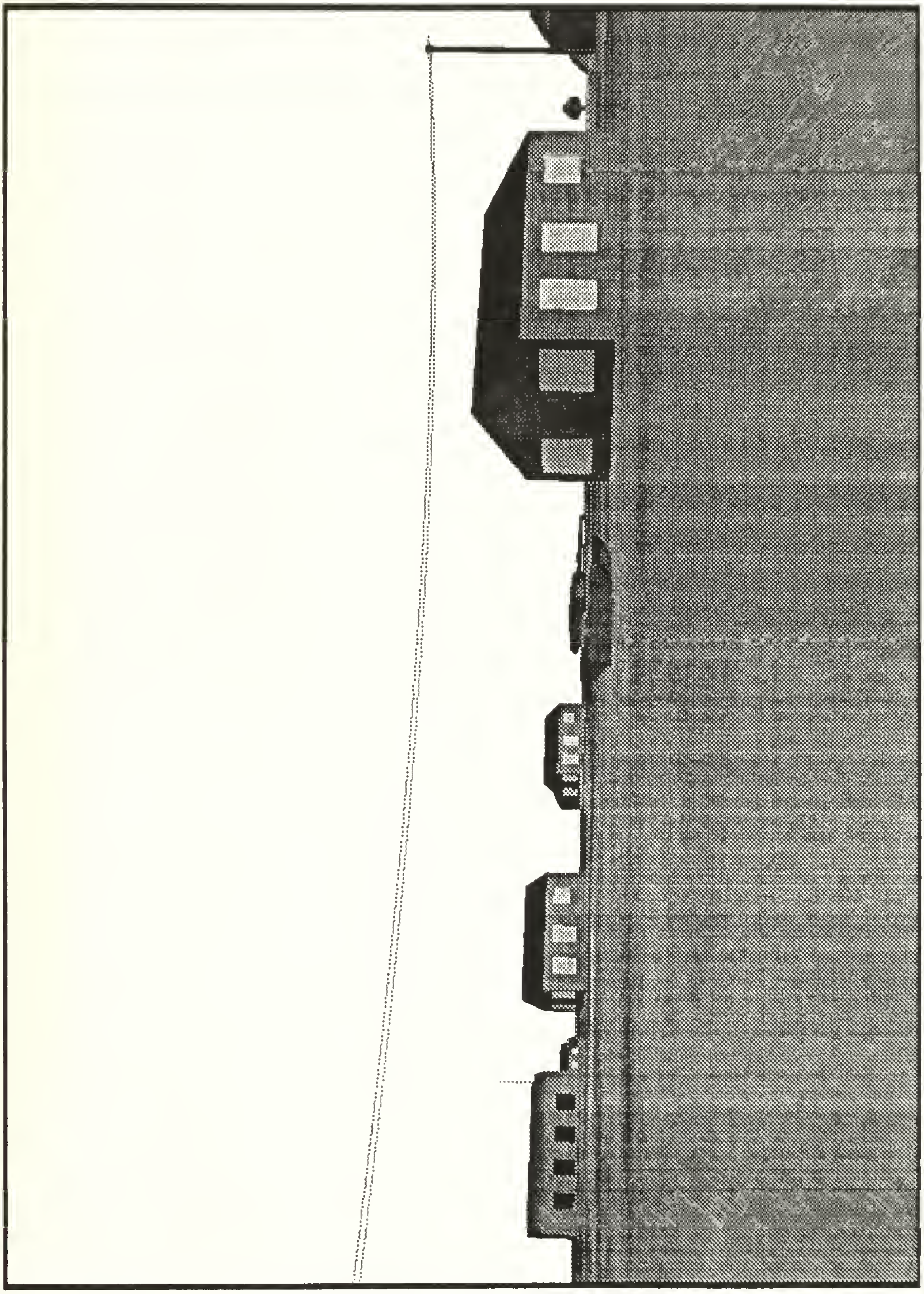


Figure 3.1. Scene from NPSNET

The dataset for Fort Hunter Liggett consists of information pertaining to the polygons composing the terrain surface, soil types (sand, water, etc.), cover (trees, bushes, etc.), natural features, man-made structures, temporal effects (munitions explosions), networks (roads, waterways, powerlines, etc.), and vehicle descriptions. The resolution of the terrain data is one point for every 125 meters. The terrain surface data consists of explicit descriptions for the polygons composing the terrain surface; that is, each polygon is described by a three-dimensional coordinate for each of its vertices. Note that this is significantly different from a system that only uses elevation data, where the coordinate location of each elevation can be derived from its relative location within the dataset. The locations of different types of trees and bushes, as well as other features such as large rocks, buildings, and roads, are also extracted from the dataset for use in NPSNET. BBN also provided their Application Programmer's Interface and this is used to extract the data. This is done off-line and the data is preprocessed before it is used in NPSNET. The way the data is manipulated and stored for use in NPSNET is discussed in Chapters Four and Five.

B. FEATURES OF NPSNET

1. Hardware

Currently NPSNET is running on different models of Silicon Graphics IRIS workstations within the NPS Graphics and Video Laboratory. These machines are a single processor IRIS 4D/30, IRIS 4D/70GT ,and 4D/310VGX; a two processor IRIS 4D/ 120GTX; and a four processor IRIS 4D/240VGX. These workstations are specifically designed to support graphics applications and contain special hardware to perform coordinate matrix transformations and clip polygons not within the field of view. Each machine contains slightly different versions of graphics hardware as well as different amounts of main memory.

2. Networking

All of the workstations running NPSNET can communicate over the NPS Computer Science Department's Ethernet Local Area Network. The machines communicate using a standardized message format. Users on different workstations can attack each other, or conduct coordinated maneuvers to specific objectives, in real-time.

3. Platform and Feature Icons

NPSNET uses descriptions of platforms and features stored in the NPS Object File Format (OFF) (Zyda, 1991a and 1991b). A set of tools have been developed at NPS to support creation and editing of these icons. A set of procedures provides access to files containing the icon descriptions and provides an interface to efficiently render them.

4. Collision Detection

Collision detection takes several forms in NPSNET: collisions between moving platforms, collisions between a platform and a fixed feature, and "collisions" between a munition fired by one platform and another platform. Currently in NPSNET one platform can fire a munition at another platform and destroy it. Detection of collisions can be computationally expensive and this is an ongoing area of research.

5. Semiautomated Forces

Inclusion of Semiautomated Forces in NPSNET is also an ongoing area of research. Currently some platforms can detect munitions fired from another vehicle and respond accordingly. Armed platforms will return fire and unarmed platforms maneuver to escape the incoming munitions. Semiautomated vehicles vary their direction of travel, based on their current speed, versus simply traveling in a straight line.

6. Scripting

NPSNET has the capability to record and playback scripts of vehicle movements. For example, one script contains vehicle movements for a coordinated amphibious landing. Work has also been done on playing back scripts from other sources.

One successful endeavor involved playing back a script of vehicle movements generated by JANUS, the Army's primary combat model.

7. Texturing

The IRIS VGX series provides texturing, a process in which a digital image is overlaid on underlying polygons. Texturing is used in NPSNET to provide more realistic terrain, and to provide a sense of motion and depth perception over that terrain. Texturing is also used to provide more realistic features on the terrain.

8. Environmental Effects and Lighting

NPSNET is able to simulate environmental effects such as clouds, smoke and haze. This effectively models the obscuration caused by the presence of these factors on a battlefield. NPSNET also uses a lighting model to produce shading in scenes.

9. Temporal Effects

Temporal effects are those that occur over a period of time. Currently NPSNET is able to model explosions as a temporal effect. On machines that support texturing, a series of digital images is progressively overlaid onto polygons to create the temporal effect of an explosion growing in intensity and then fading over a short period of time.

IV. TERRAIN PAGING IN NPSNET

A. DATASET AND MEMORY SIZE

The dataset used for the 50 by 50 kilometer area used in NPSNET contains 79 megabytes of data in binary format for the terrain polygon descriptions alone. This size increases to 147 megabytes when three additional, lower resolution descriptions of the same terrain are generated. Of the machines currently running NPSNET, the one with the most main memory has only 64 megabytes. All of the data cannot be loaded into a data structure at one time. If the simulator is to have all of this terrain available, a method is needed to allow part of the data to be in memory at a time, and page terrain in and out of main memory as it is needed.

B. DATASET PREPROCESSING

The original SDIS dataset stores 64 square kilometers of terrain data in one file. These files are too large to use this as a basis for paging terrain; their granularity is too coarse in terms of area and size of file. The one kilometer standard of the military "grid square" and the 125 meter resolution of the dataset leads to dividing the terrain data into files where one square kilometer of terrain data is stored in one file. This results in 2500 files, each file describing the terrain for one square kilometer of the 50 by 50 kilometer dataset. Dividing up the data in this way facilitates paging one kilometer strips of terrain, composed of one kilometer squares, in and out of memory in each of the four cardinal directions.

In order to prepare the data, the dataset is processed off-line. Each square kilometer's worth of data is initially stored in a corresponding text file. The file is subsequently

converted from text to binary format to facilitate the use of block reads for bringing the data into a data structure. Storing the number of polygon descriptions at the head of the file facilitates the allocation of memory for the data prior to it being read into memory (every polygon description is the same size).

The implementation of a hierarchical data structure is separate from terrain paging in this implementation. Different data structures could be used to describe the terrain. The final form of the dataset used by NPSNET is 2500 binary files; each containing a multiple resolution description of the terrain for one square kilometer, stored as a heap-sorted quadtree. The form of the data structure in main memory is an array of quadtrees. The implementation of the quadtree structure is discussed in Chapter VI.

C. ACTIVE AREA SIZE

Considerations for the memory sizes of the available workstations, frame rates, the required field of view, and desired range of views suggest using a 16 by 16 kilometer active area of terrain. This amount of terrain data will be in memory at any time and available for rendering. Sixteen kilometers allows seven kilometers of terrain in opposite directions for immediate rendering and one kilometer in each direction as a buffer to ensure terrain is fully paged in before attempting to render it. This buffer zone is necessary because terrain paging is done in parallel. On multiple processor machines the simulator does not wait for additional terrain to be paged in, an updating process is unblocked and motion continues. By coordinating the number of simultaneous updates with the size of the buffer zone, the problem of attempting to access data that is not yet in memory is avoided. Figure 4.1 depicts the concept of a dynamic active area of terrain within the larger area for which data is available.

D. ARRAY INDEXING

Initial developmental versions of the code to accomplish terrain paging in NPSNET used a 16 by 16 array corresponding to one square kilometer areas of terrain. Each element of the array contained a data structure of the terrain data for one square kilometer. The location of the driven vehicle determined the current center of the active area of the terrain and the values of the indices of this square of terrain were maintained. By using modulo 16 arithmetic, the elements of the array could be addressed appropriately. The expense of modulo arithmetic became apparent and the data structures and the code were modified.

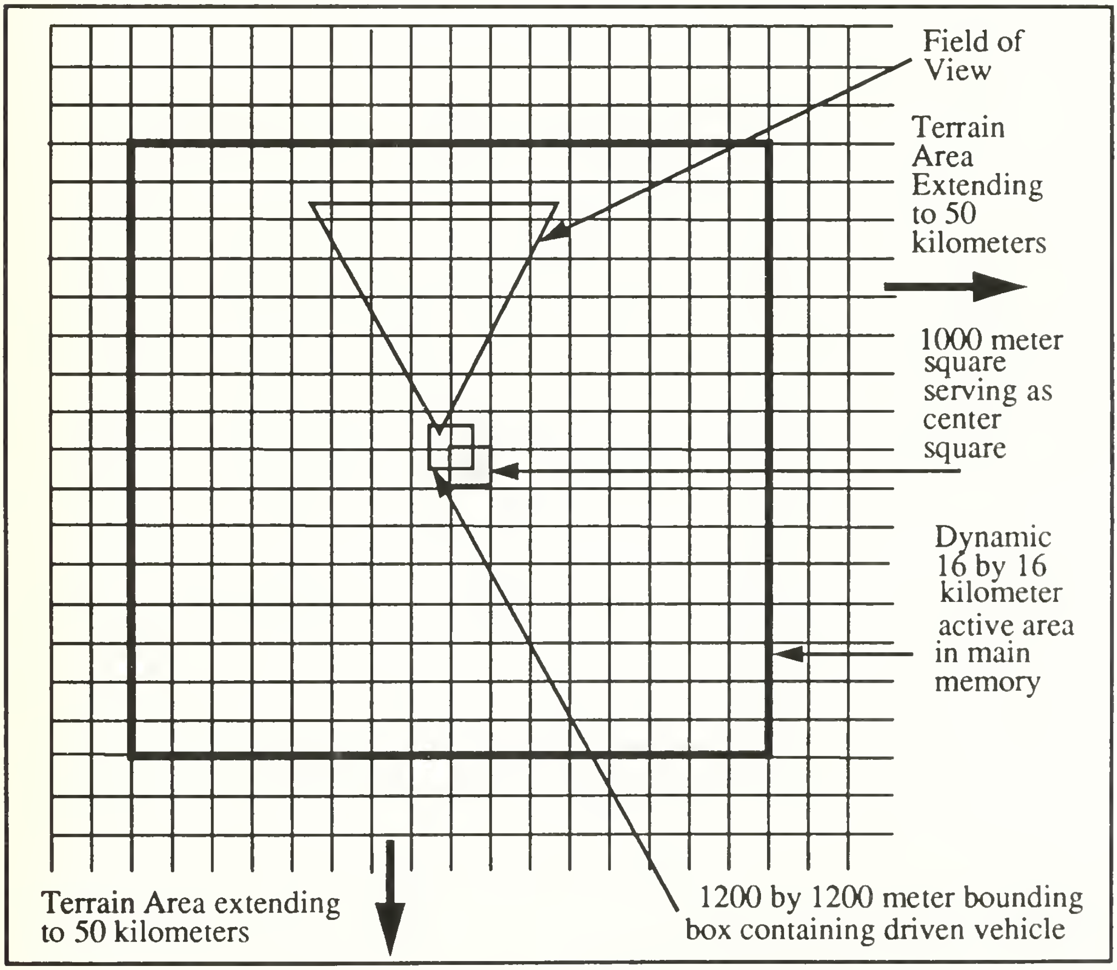


Figure 4.1. Active Area of Terrain in NPSNET

The current version uses a 50 by 50 array as the top level data structure, but only a 16 by 16 part of this array contains terrain data at any one time. The values of the indices of the center one kilometer square of terrain are maintained as the notional center of the active area of terrain. Before data for a strip of 16 square kilometers is paged in to the east, the memory for the western 16 square kilometers is freed. This limits the amount of data in memory at any one time. The terrain data files vary in size based upon the roads in each area, but they average 15 kilobytes. This means approximately four megabytes of main memory are needed for terrain data at any given time, compared to 147 megabytes for the entire dataset.

E. TERRAIN PAGING ALGORITHM

When the simulator is initialized, data for a 16 by 16 kilometer area is loaded into the appropriate elements of a 50 by 50 array. This 16 by 16 kilometer active area is centered about the location of the vehicle the user is occupying. This eliminates the need for modulo arithmetic in calculating array indices. If the driven vehicle reaches the bounding box in any direction, memory space is freed in one direction and terrain is paged in the opposite direction. The center square changes, and the notional box surrounding the driven vehicle moves as shown in Figure 4.2.

The size of the box surrounding the driven vehicle relative to the basis by which the terrain data is divided is important. In this case, the bounding box is 1200 by 1200 meters and the data is divided on a 1000 meter basis. Use of a 1000 by 1000 meter square bounding box presents the possibility of thrashing. The driven vehicle could continually move back and forth across the bounding box. The edges of the new and old bounding box would coincide; there would be no overlap between the two boxes. Use of a 1200 by 1200 meter bounding box requires the driven vehicle to change directions and move 200 meters in the opposite direction before the terrain just paged in would be deleted. This concept is

illustrated in Figure 4.2. A 200 meter overlap may not be enough in some cases where vehicles or aircraft rapidly and continuously reverse directions, but the size of the bounding box could be increased even more to increase the overlap distance between the bounding boxes.

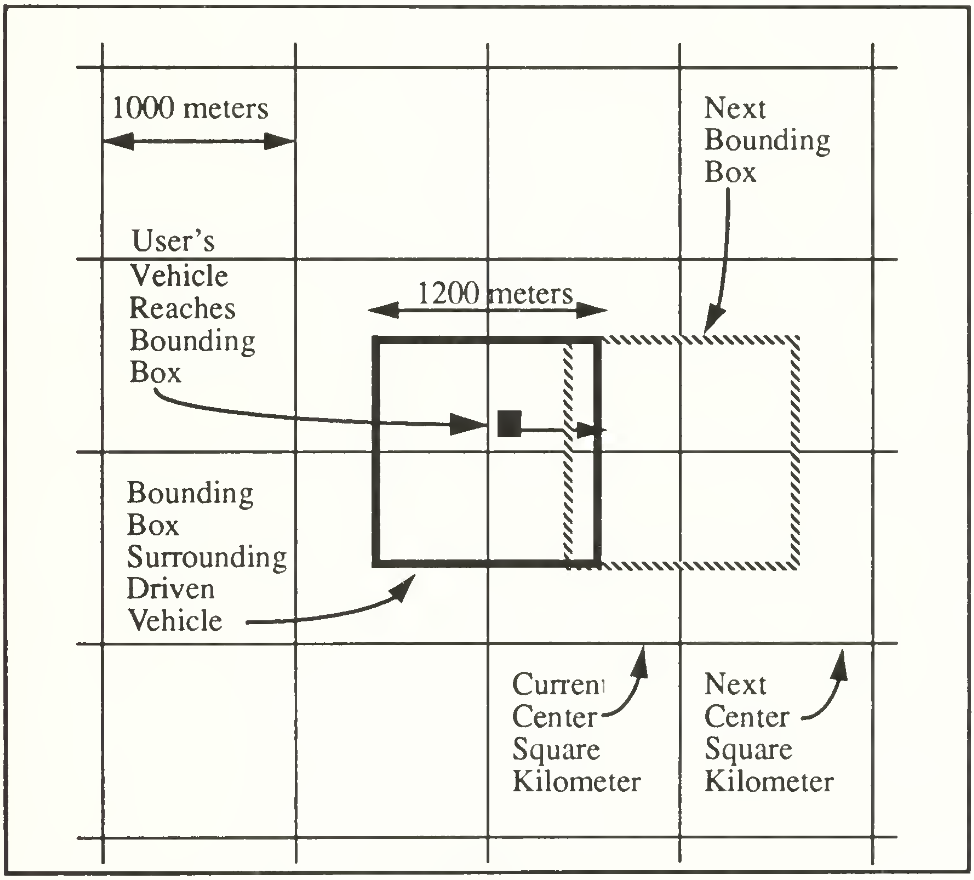


Figure 4.2. Boundaries for Terrain Paging in NPSNET

V. HIERARCHICAL DATA STRUCTURES FOR VISUAL SIMULATORS

The use of hierarchical data structures in computer graphics provides two main benefits. The first is that they assist in quickly determining what is and isn't visible in a particular view of the data. The second is that hierarchical data structures facilitate implementing multiple resolution versions of data. Both of these benefits combine in reducing the total number of polygons rendered in each individual frame, thereby allowing a faster frame rate and improving the overall quality of the simulation.

A. QUADTREES

Quadtrees are a type of hierarchical data structure based on recursive subdivision of an image or data. The general idea behind quadtrees is that each level of the tree provides a more detailed description of an image or data. Nodes are either internal and have four children or are terminal and have no children. Quadtrees were initially applied in the field of computer graphics to implement hidden line and surface algorithms and as a way to store pixel data for two-dimensional images (Samet, 1990a, pp. 10-15). They can be used to reduce storage space for data or to provide multiple resolution versions of data.

Quadtrees can be implemented in number of ways. The most straightforward is a pointer implementation, in which each node points to its own data and to its four children (unless it is a terminal node). As with any tree based structure, quadtrees can also be conveniently represented as a heap. In a heap-sorted quadtree, the offset from a parent to its four children are 4n+1, 4n+2, 4n+3, and 4n+4 where n is the level of the parent (the root node is level 0). Quadtrees can also be represented as a collection of leaf nodes. Various encoding techniques to describe the location of the nodes within the tree have been

developed. One of these involves the use of quaternary (base four) integers to record the path from the root node to each other node (Gargantini, 1982, pp. 905-907). A number of algorithms to support such operations as search, neighbor finding, and nearest node in quadtrees have also been developed. These can be found in *The Design and Analysis of Spatial Data Structures* (Samet, 1990b, pp. 57-110).

1. Quadtrees in Two-Dimensional Applications

Quadtrees are well suited for processing two-dimensional images since the image can be described by a two-dimensional array of pixel data. The advantage of using a quadtree for this type of application is that the quadtree can reduce the amount of memory needed to store the image. A quadtree can effectively combine areas of an image in which adjacent pixels are described by the same values as shown in Figure 5.1. In this type of approach, where the goal is to reduce storage space, only the terminal nodes would contain pixel data. Note that some nodes contain information for adjacent groups of pixels; these groups contain a number of pixels that is a power of four.

The approach of using a quadtree to create multiple resolution version of an image is slightly different. In this type of approach every node would contain data as shown in Figure 5.2. This example shows a quadtree containing the description of an image in four different resolution versions. More memory storage space is required for this multiple resolution version of the image, but note that the quadtree is still able to take advantage of areas of the image that only require descriptions less detailed than that provided by the highest resolution. A full quadtree would contain data for all pixels up to the highest resolution; again the internal nodes would be used to provide multiple resolution descriptions of the image. A full quadtree representation would not reduce storage space for the image data.

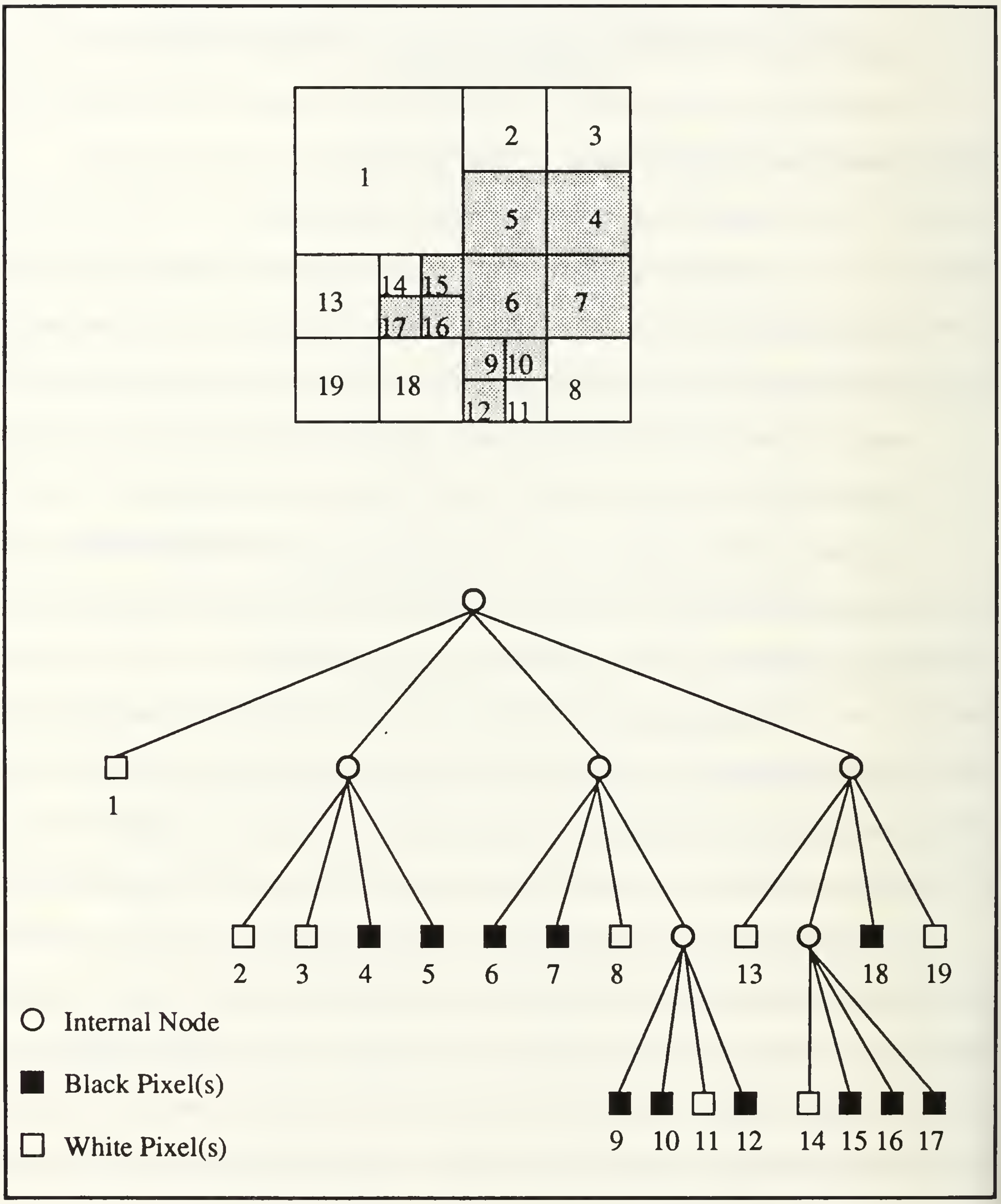


Figure 5.1. Quadtree Representation of 2D Image

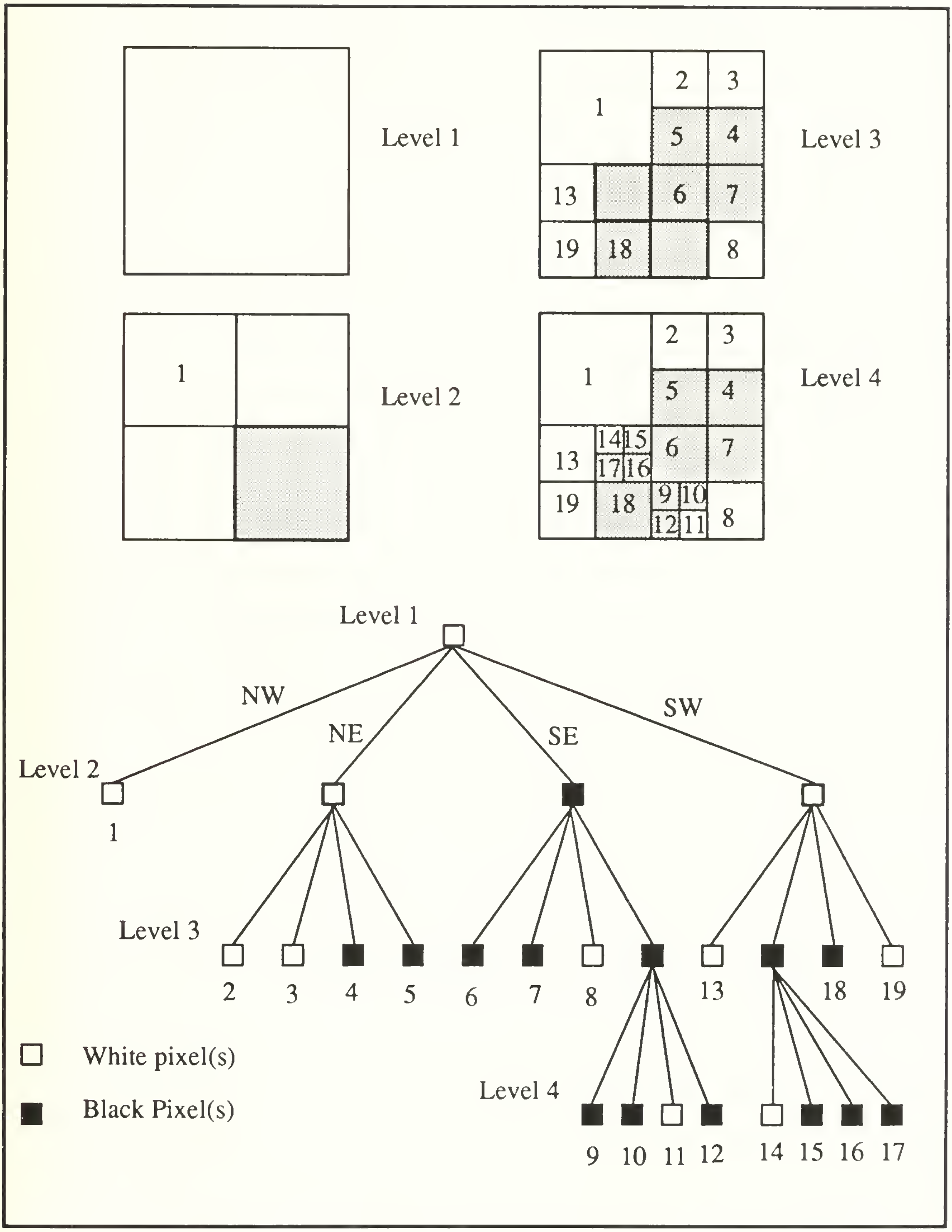


Figure 5.2. Quadtree Containing Multiple Resolution Description of 2D Image

2. Quadtrees in Two-and-One-Half-Dimensional Applications

Quadtrees can be extended from two-dimensional applications to applications involving two-and-one-half-dimensional surfaces such as terrain. The nodes of the quadtree provide a spatial index to polygonal descriptions for particular surface areas (Samet, 1990a, p. 11). The data could also be in the form of surface patches for the corresponding areas of terrain (Schachter, 1983, p. 112). Quadtrees in these types of applications provide the same benefits as in two dimensional applications: they provide the potential to reduce storage space and they provide multiple resolution descriptions of the surface data.

Quadtrees provide the capability of describing two-and-one-half-dimensional surface data only to the level of detail necessary to describe the surface. Adaptive refinement (DeHaemer, 1990) is a method by which a surface is recursively described in successively finer levels of detail until the description of the surface lies within a certain tolerance. A quadtree supporting this type of description is shown in Figure 5.3. Note that in this particular structure only the terminal nodes provide access to data; therefore, no multiple resolution versions of the object are provided.

Another approach is to describe all areas of the surface to the highest resolution possible, even if parts of the surface need not be described in higher detail. This produces a full quadtree. Providing access to data at every node provides multiple resolution descriptions of the surface. A surface in different levels of detail is shown in Figure 5.4. A quadtree used to describe a surface in this manner is shown in Figure 5.5.

One implementation uses quadtrees to store digital elevation data (Air Force Wright Laboratories, 1986). Quadtrees are used in this application to store the average elevation value for an area corresponding to a node in the quadtree. A tolerance is used to determine if elevation values need to be stored at the lower level (higher resolution) nodes of the quadtree.

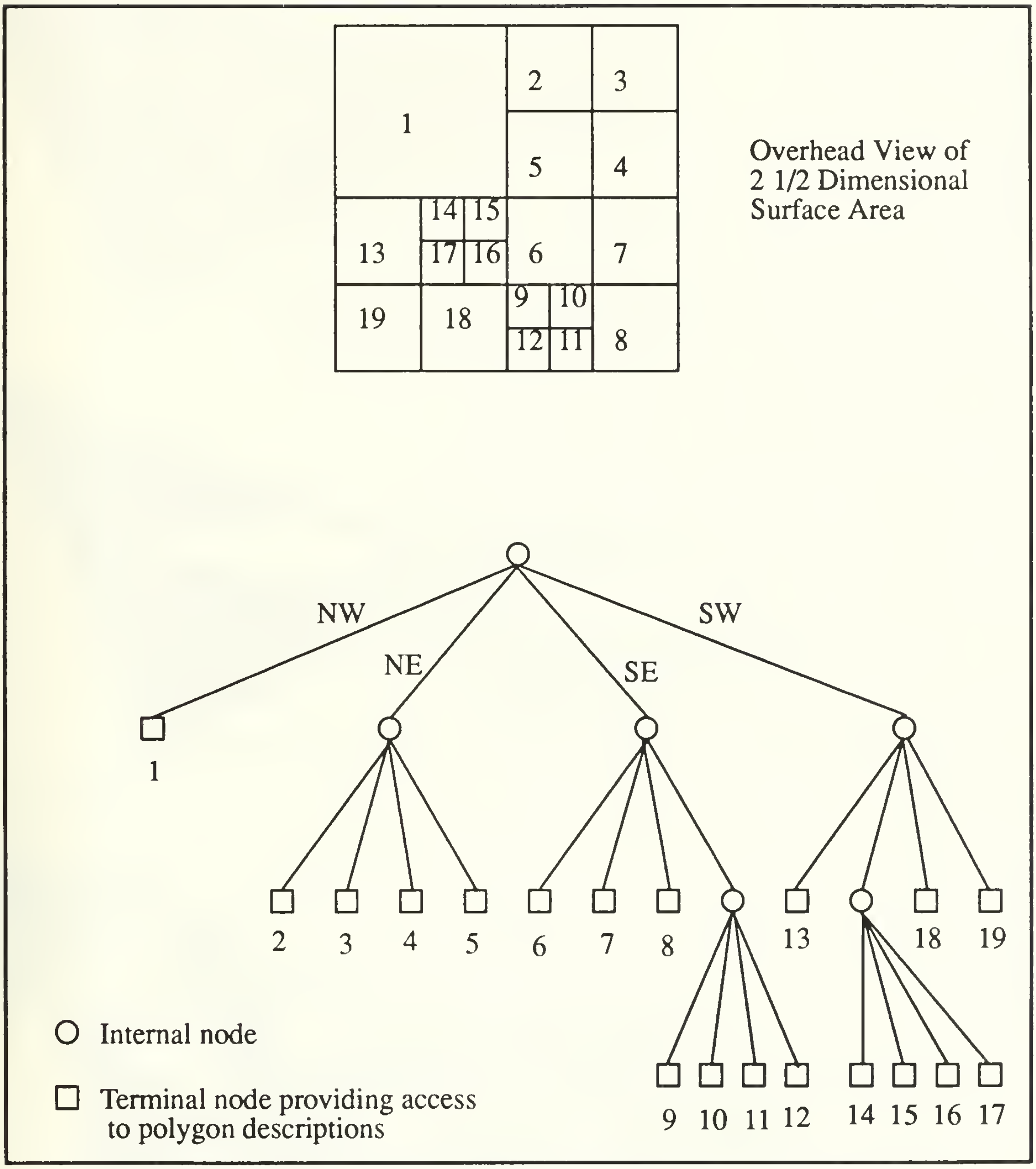


Figure 5.3 Quadtree Providing Spatial Index into 2 1/2 Dimensional Area

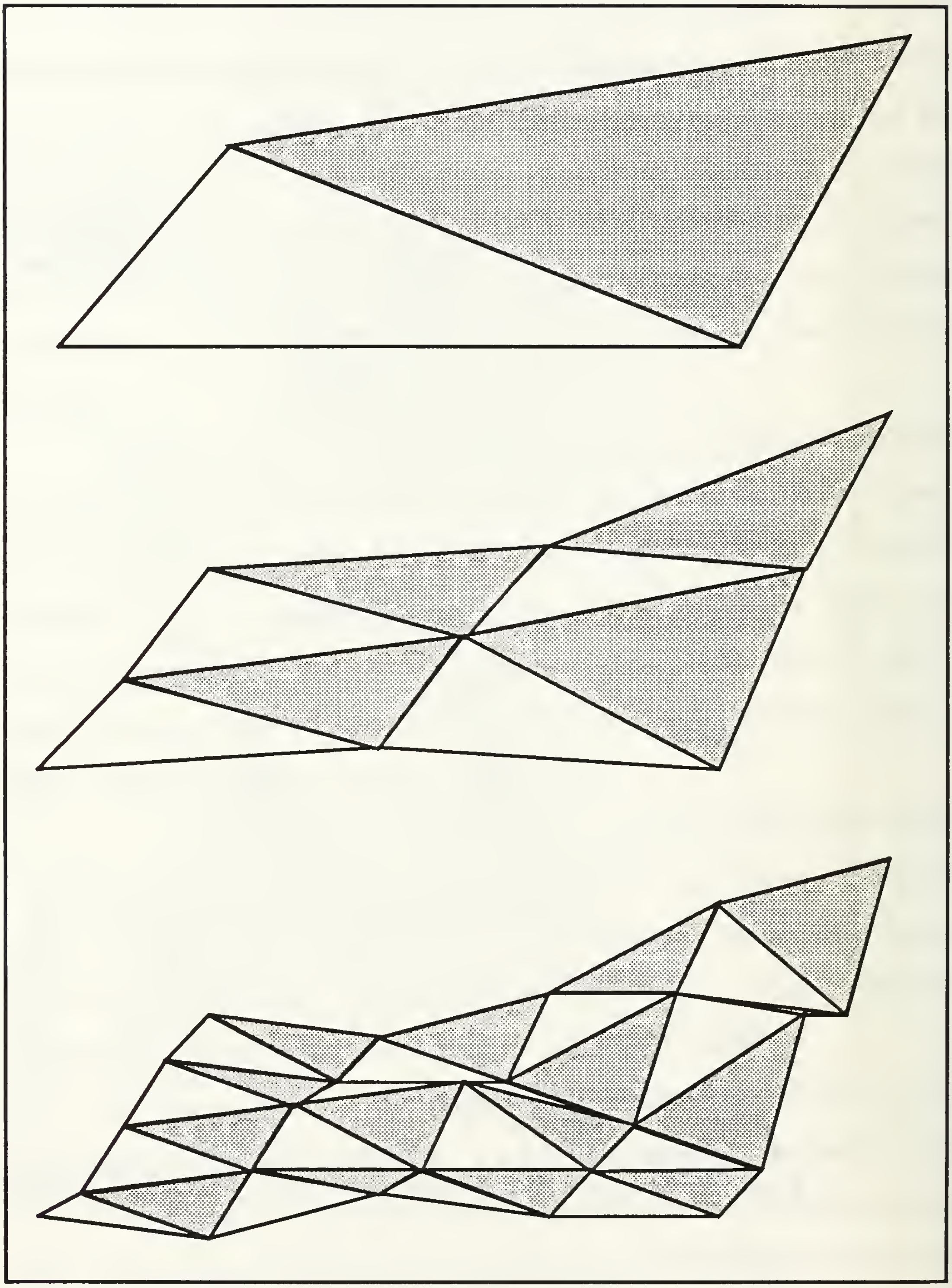


Figure 5.4. Multiple Resolution Descriptions of 2 1/2 Dimensional Surface

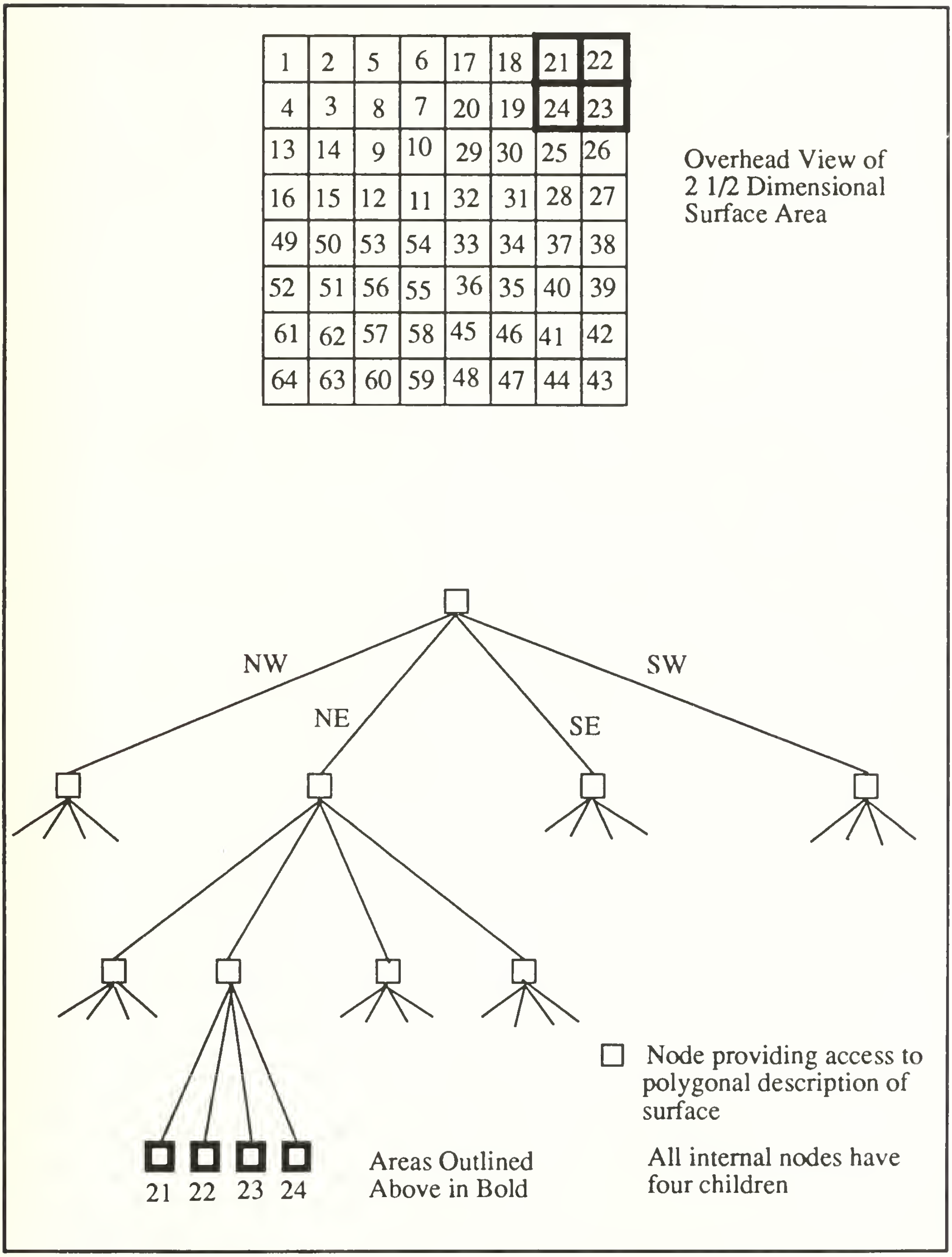


Figure 5.5. Full Quadtree Providing Spatial Index to Multiple Resolution Description of a 2 1/2 Dimensional Surface

B. OCTREES

Octrees are an extension of quadtrees and are used to store three-dimensional volume data. In an octree each internal node has eight children and terminal nodes have no children. An example of an octree and a node numbering scheme is shown in Figure 5.6. An octree is developed by recursive decomposition of a three dimensional object; that is, each octant can be further divided into its own component octants.

An octree can record which subvolumes or voxels (VOlume piXEL) are occupied by an object. The octree description then provides an approximation of the object, the accuracy determined by the resolution to which the object is decomposed. Using an octree to record the voxels composing an object is shown in Figures 5.7 and 5.8. Note how an octree used in this manner records the solid nature of an object.

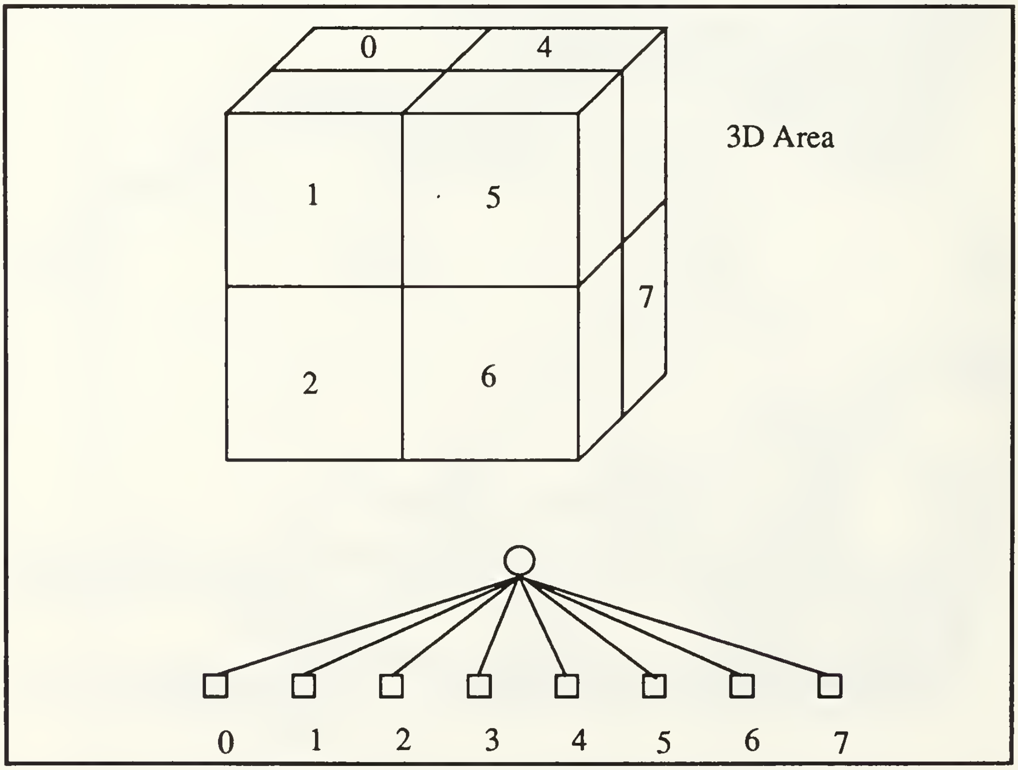


Figure 5.6. An Octree Numbering Scheme

Octrees can also provide a spatial index into three-dimensional data surface. Each node would provide access to the polygons in the particular area of three-space associated with the node. In this scheme not all children of a node may contain data. Like quadtrees, octrees can be used to provide multiple resolution descriptions of a three dimensional object. This applies to both octrees describing the voxels composing a solid and to octrees containing polygonal descriptions for a three dimensional surface.

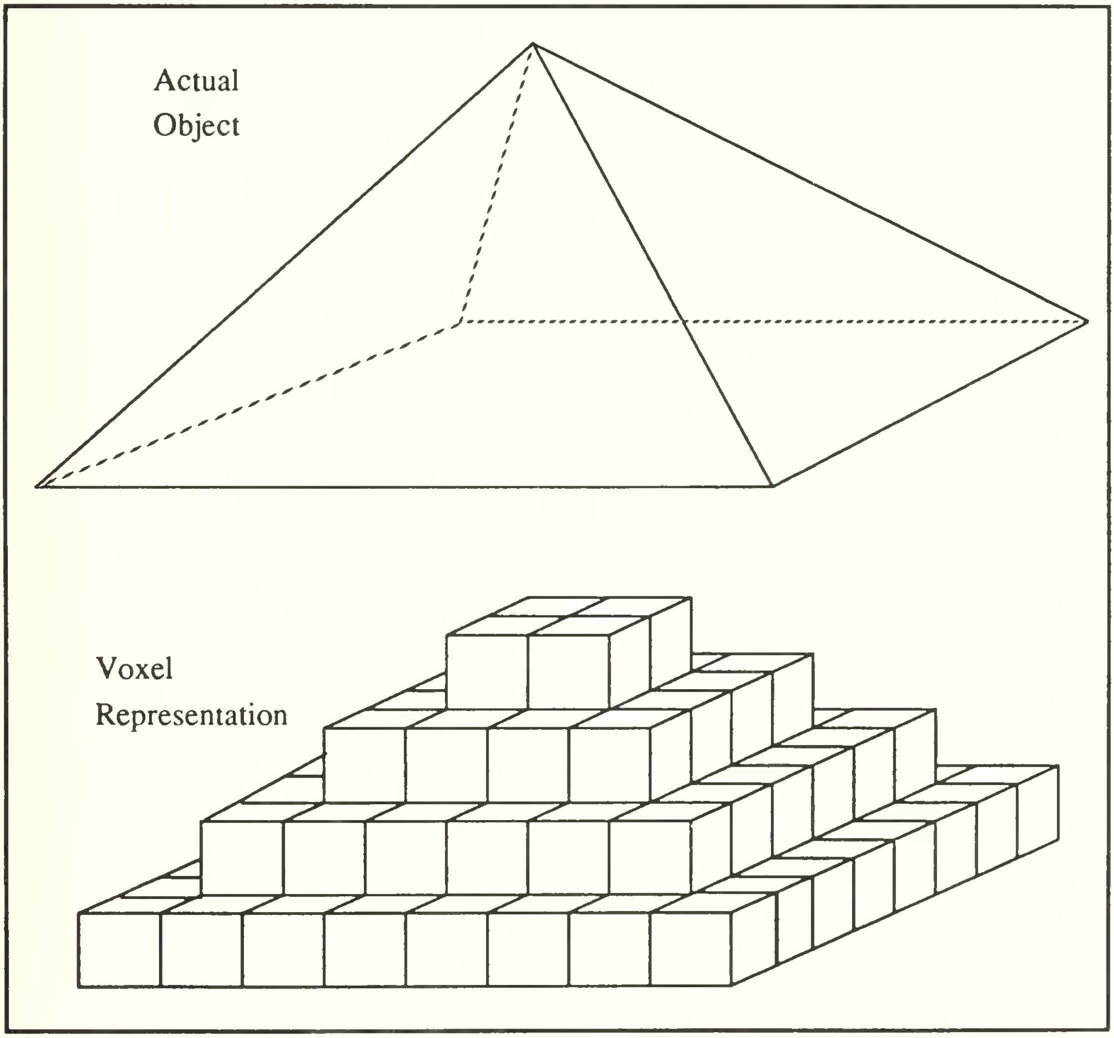


Figure 5.7. Voxel Representation of a Solid Object

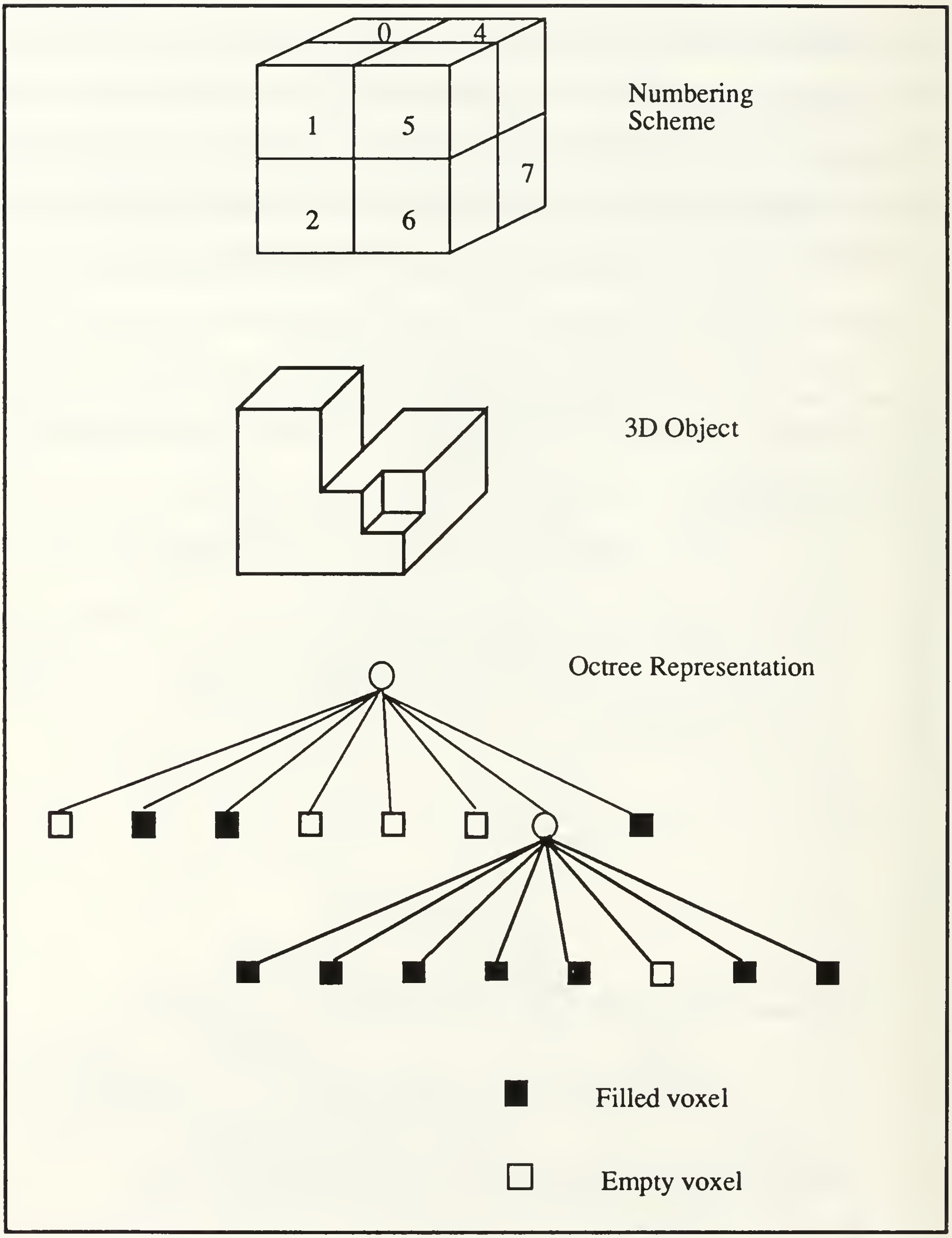


Figure 5.8. Octree Encoding of an Object Represented by Voxels

C. BINARY SPACE PARTITIONING TREES

Binary Space Partitioning Trees (BSP Trees) recursively divide an object or space using dividing planes (Fuchs, Kedem, and Naylor, 1980). While the octree effectively does this based on a regular pattern, the dividing planes used in BSP trees may be arbitrary planes in three-space. Figure 5.9 shows a BSP tree representation of an object in two dimensions using dividing lines. In this example, the dividing lines are still perpendicular or parallel, but this does not have to be the case. The extension to three dimensional objects and dividing planes in three-space is straightforward.

The use of BSP trees in architectural applications is an area of active research (Airey, 1990). The floors and walls of a building serve as ready made dividing planes used to partition the space of the building. A Potentially Visible Set (PVS) of polygons can be calculated and associated with each cell created by the partitioning (Airey, 1990, p. 40). This translates into what a viewer could see if he was in a particular room, including parts of other rooms visible through doors or windows. Careful selection of which planes to use in dividing the data can effect the compactness and efficiency of the partitioning (Airey, 1990, pp.16-20).

The partitioning can also be used to divide the data so that all of it need not be loaded into memory at the same time. For example, in walking through a building, a viewer would move between rooms often and between floors infrequently. Data pertaining to the building could be paged in and out based on the floors of the building. Perhaps only enough memory is available to have data pertaining to three floors in memory and available for rendering at any one time. In this case the data pertaining to the floor the user is on, and to the floors above and below could be in memory. As the user moves up a floor, the data pertaining to the lowest floor is no longer needed and data pertaining to a new floor at a higher level could now occupy that memory space.

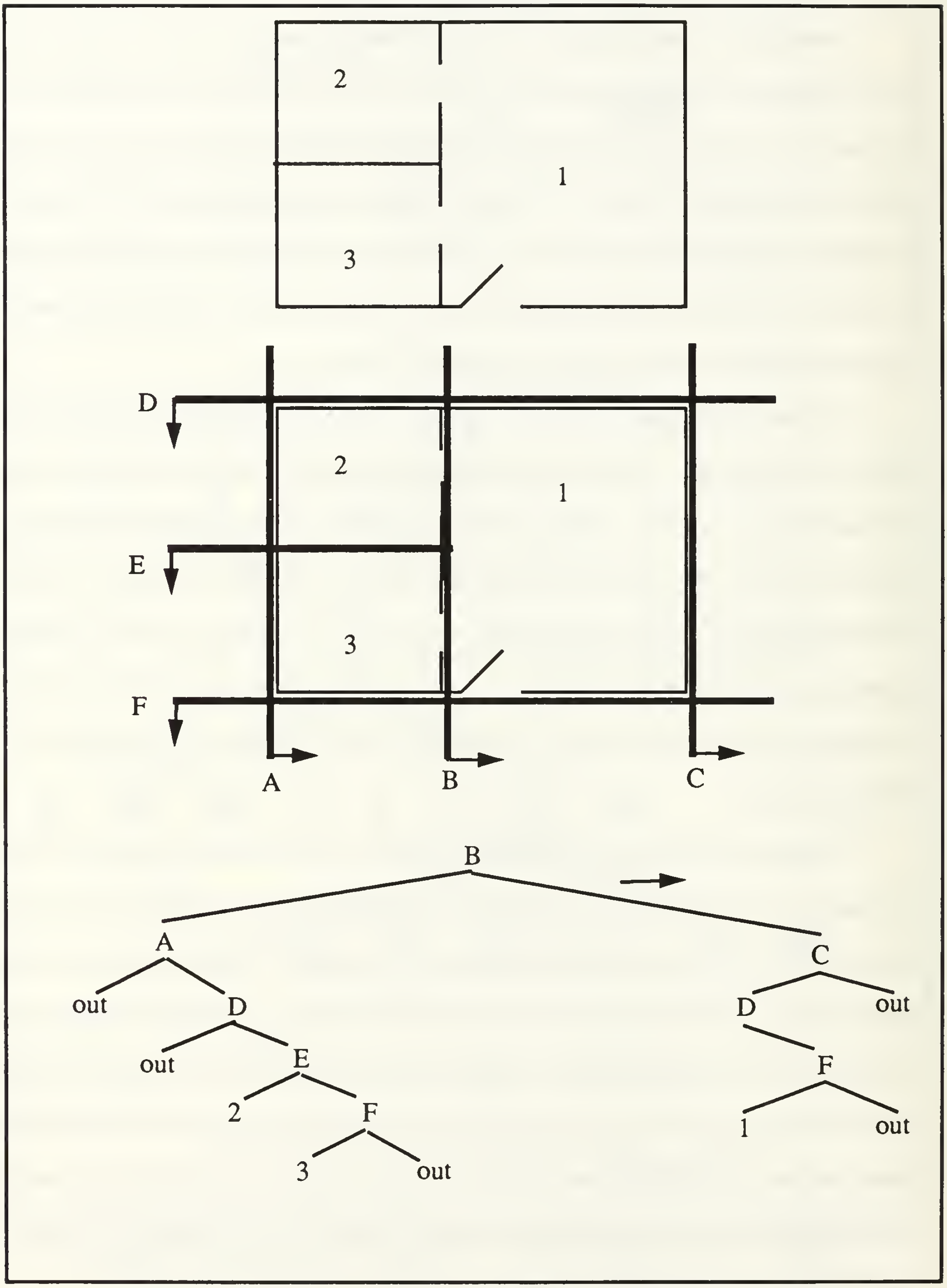


Figure 5.9. BSP Tree Representation of a Building Floor Plan

D. SELECTION OF A DATA STRUCTURE FOR NPSNET

A quadtree structure is chosen for implementation in NPSNET over other data structure for several reasons. The nature of the terrain data leads to using a data structure that can take advantage of the regularity of the data points. Nature does not provide terrain features that can serve as convenient dividing planes for partitioning data based on some type of BSP Tree. Octrees are normally associated with three-dimensional applications, but NPSNET is only concerned with the terrain surface and is a two-and-one-half-dimensional application. A quadtree can provide a spatial index into the data and also requires less in the terms storage requirements than an equivalent octree representation.

VI. IMPLEMENTATION OF QUADTREES IN NPSNET

A. DATASET PREPROCESSING

In order to implement quadtrees in NPSNET the dataset is preprocessed. The dataset used in NPSNET provides a description of the polygons that comprise the underlying terrain surface based on data points every 125 meters. Due to the considerations for paging terrain in and out of memory, separate files were established for each square kilometer of terrain as discussed in Chapter IV. Two main concerns guide dataset preprocessing: generating three additional versions of the data in successively lower resolutions and storing the data in a manner that facilitates fast paging.

1. Terrain Polygon Descriptions

The polygons forming the terrain surface are described by the set of attributes shown in Appendix A. The allowable values are also shown. The set of attributes are the same no matter what resolution of terrain a particular polygon describes. No attribute indicating what resolution level a particular polygon describes is necessary, since this can be derived from the polygon description's relative location within the file. Likewise, the descriptions of polygons that form features like roads are described by the same set of attributes. Other features and vehicles displayed in NPSNET are described using the NPS Object File Format and are displayed using a set of procedures designed for that format. These object descriptions are stored in files separate from the terrain data.

2. Generating Lower Resolution Versions of the Terrain Surface

The general idea behind forming lower resolution versions of terrain is simple. Triangles describing four 125 by 125 meter areas are combined to form two triangles describing a 250 by 250 meter area as shown in Figure 6.1. Likewise, sixteen 125 by 125

meter areas are combined to form two triangles describing a 500 by 500 meter area, and similarly for forming a 1000 by 1000 meter area. Examples of this are shown in Figures 6.2 and 6.3. The generation of lower resolution descriptions concerns the underlying terrain surface only, and is not concerned with features that lie on top of that surface such as roads. In the current version of NPSNET, roads are only rendered on high resolution terrain.

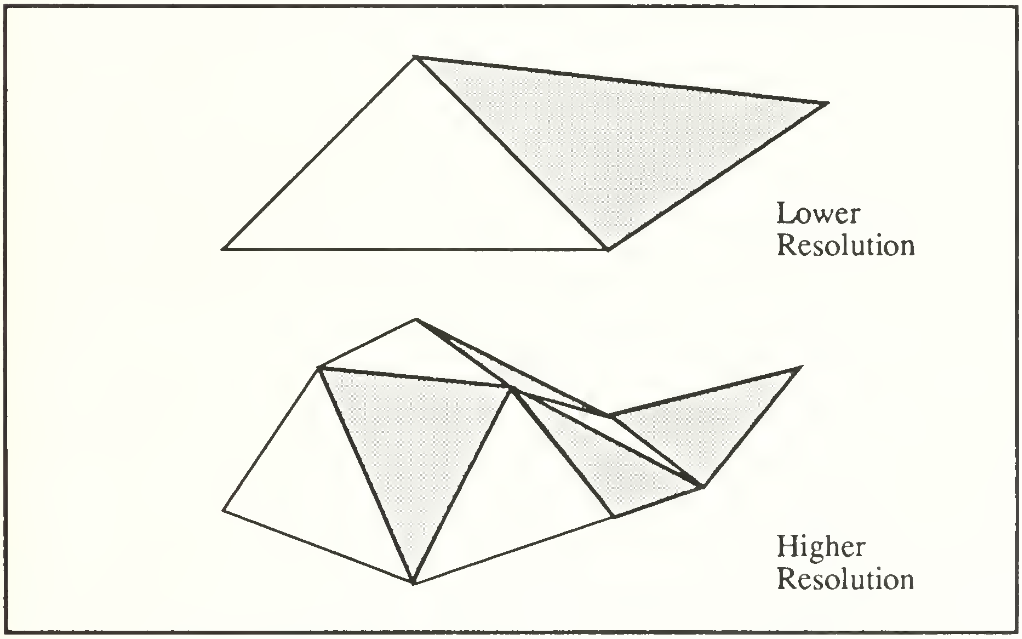


Figure 6.1. Multiple Resolution Terrain Description

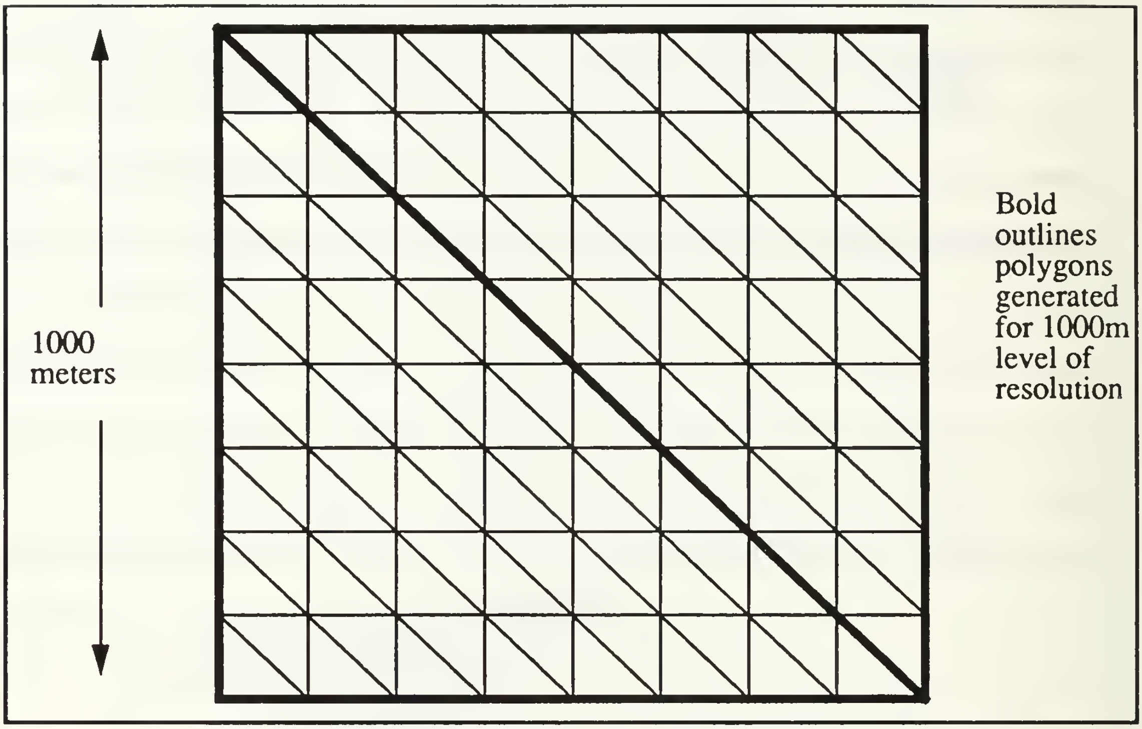


Figure 6.2 Generating 1000 Meter Level of Resolution

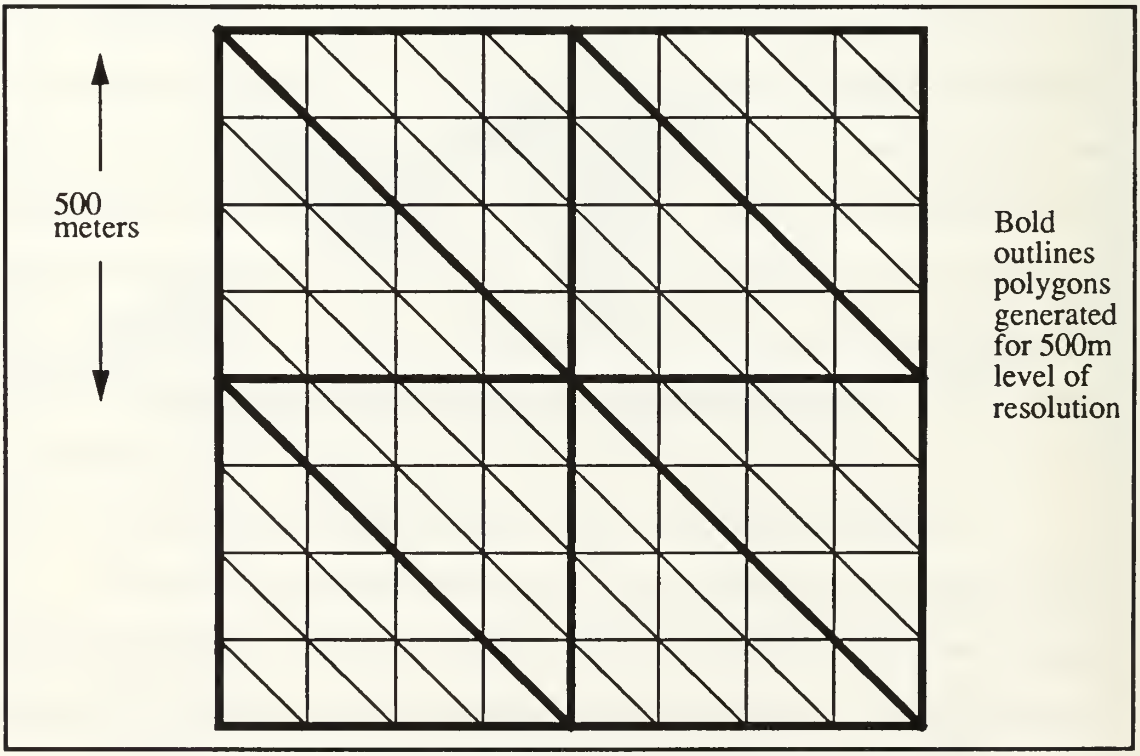


Figure 6.3. Generating 500 Meter Level of Resolution

The lower resolutions of the terrain surface for NPSNET are generated one square kilometer at a time The data for a square kilometer is loaded into an array. The size of the array is eight by eight, corresponding to the 125 by 125 meter areas of one square kilometer. The elements of this array are pointers to lists of terrain surface polygon descriptions. Each list contains all of the polygons for a particular 125 by 125 meter area. Most 125 by 125 meter areas can be described with two triangles; however, other areas may contain "micro-terrain" and require more triangles. Micro-terrain is used to more accurately describe some areas, such as shorelines around small lakes and ponds. 125 by 125 meter areas which contain roads will also have polygon descriptions for these surfaces.

The algorithm is recursive, generating the lowest resolution version of the terrain first (the 1000 meter resolution) and progressing down to the 250 meter resolution. As the lower resolution versions of the terrain are generated, they are written to a text file. The highest resolution version of the terrain (the original 125 meter resolution which already exists) is simply written to the text file at the appropriate times. This is shown in Figure 6.4.

3. Selecting Terrain Color at Lower Resolution Levels

The color for a lower resolution terrain polygon is simply the majority color of the highest resolution polygons contained in the same area. The algorithm for selecting color is shown in Figure 6.4.

One problem in selecting terrain polygon color is apparent near shorelines. The problem occurs at cliffs along the coast. In certain situations, the majority of the underlying polygons are water, but the elevations of the corners of the resulting lower resolution triangle vary greatly. This causes the generation of a "slope" of water in the lower resolution description. This problem is solved by checking the elevations of the corners of triangles whose majority color is water. If all of the corner elevations are not within a certain tolerance of each other, the color of the triangle is set to soil color. Small

discrepancies within the dataset, such as some elevation points on the ocean being over one meter (versus all being sealevel -- 0 meters elevation), requires the use of a tolerance rather than just checking the equality all corner elevations are equal. This technique still results in some lower resolution polygons that represent water, yet have corner elevations that vary slightly; however, the visual effect is not distracting.

GENERATING LOWER RESOLUTION TERRAIN

```
if this level >= 250 meter resolution
        select color for upper triangle
        select color for lower triangle
        generate four fill polygons and write them to file
        generate two lower resolution polygons and write them to file
        generate next higher resolution for four subareas
else
        write 125 meter level polygons for this area to file
end if
```

SELECTING COLOR FOR LOWER RESOLUTION TERRAIN

```
determine if majority color of underlying high resolution polygons is
   soil or water
set color to majority color
if majority color is water then
        if elevations of any two corners of new polygon differ by more
                      than tolerance then
                set color to soil color
        end if
end if
```

Figure 6.4. Algorithms for Generating Lower Resolution Terrain

Fill polygons to fill gaps in the terrain surface created at the boundaries of terrain rendered at different resolutions are also generated as part of this process. The need for fill polygons is illustrated in Figure 6.5. Each set of two triangles describing a square area of terrain at the 250, 500, and 1000 meter resolutions has four fill polygons associated with it--one on the North, South, East and West. A polygon with the same vertices will appear in two locations in the new dataset. For example, in two adjoining 500 by 500 meter areas. The vertices of these two fill polygons will be the same; however, the colors of the fill polygons may be different. The colors for these fill polygons are selected and their normals are calculated during this process. Determining which fill polygons need to be rendered in each frame is discussed in Chapter VII.

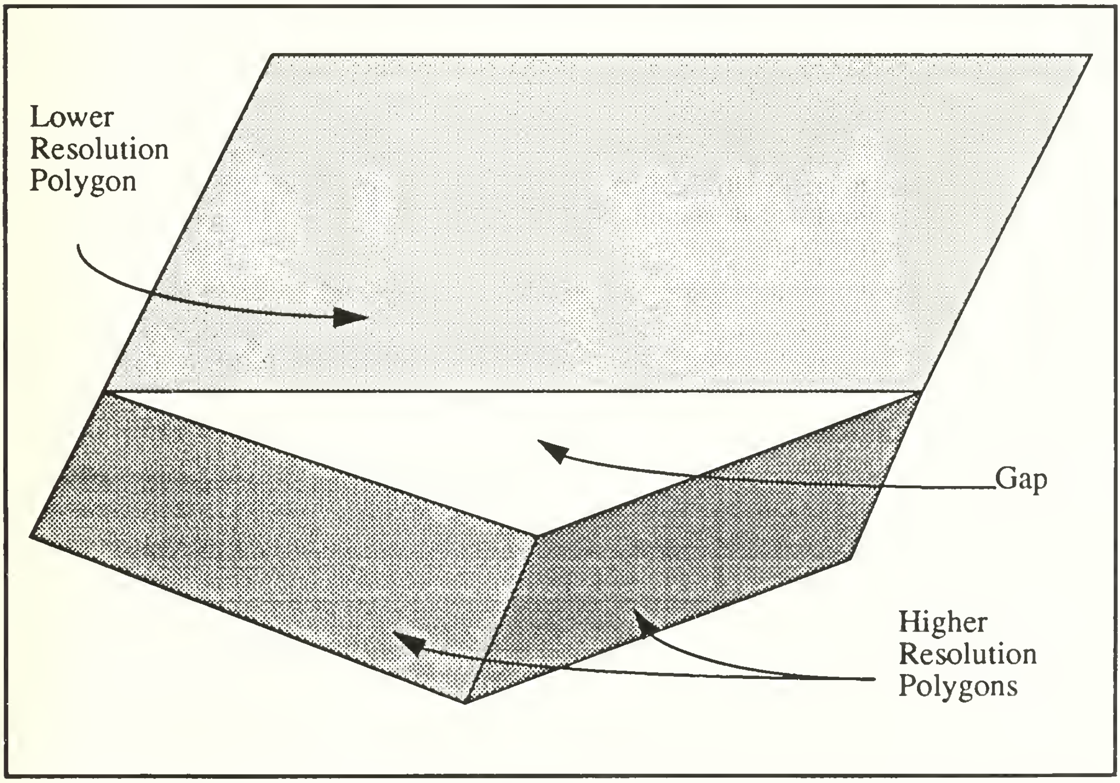


Figure 6.5. Gap Created at Boundary of Resolution Levels

B. TERRAIN DATAFILES IN NPSNET

1. File Numbering Scheme

The numbering scheme for files containing multiple resolution terrain data is depicted in Figure 6.6. The modified dataset contains 2500 files each describing one square kilometer of terrain surface in four levels of resolution.

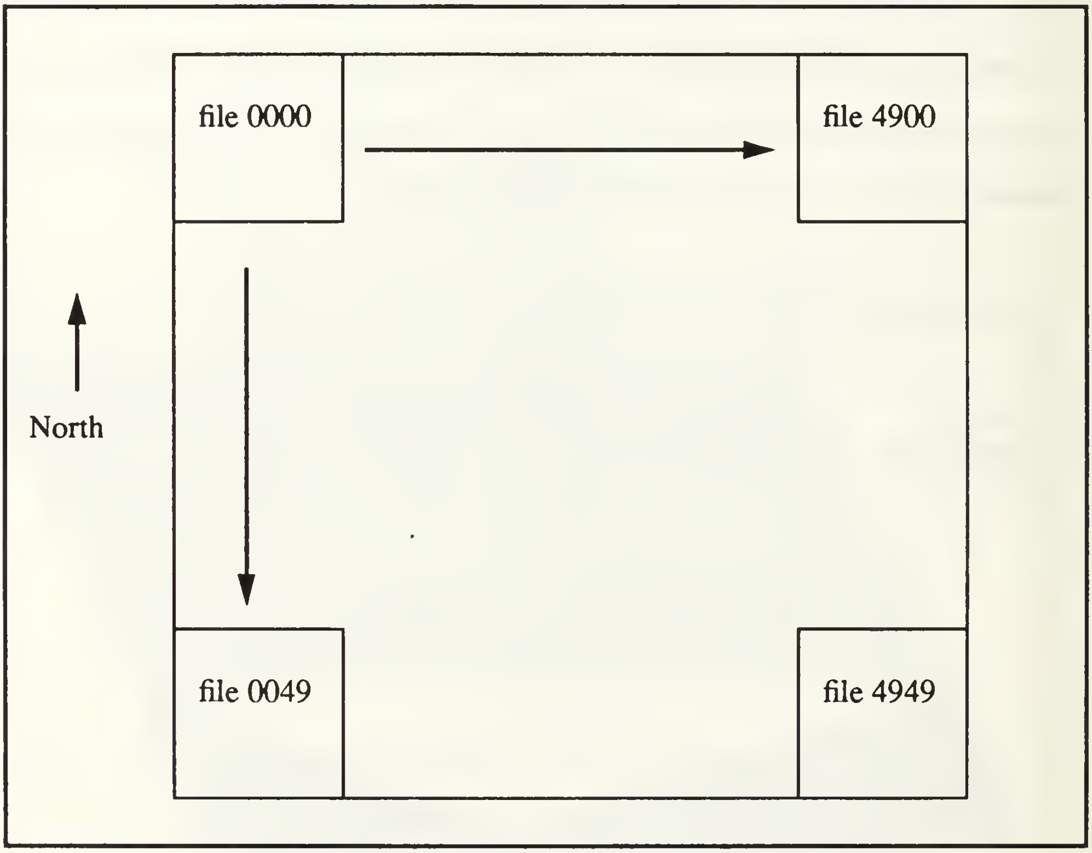


Figure 6.6. Terrain Datafile Numbering Scheme in NPSNET

2. File Format

The format of the binary terrain data files used by NPSNET is shown in Appendix B. The format is designed for fast access using the C function *fread()*. The files containing the multiple resolution versions of the terrain data are much larger than the original files. The files containing the original version of the data in binary form for the

entire 50 by 50 kilometer area total 79 megabytes. The files containing the multiple resolution version of the data total 147 megabytes. The additional size is due not only to the multiple resolution descriptions of the terrain, but also due to the addition of fill polygons required at the boundaries of terrain rendered at different resolutions. Both of the file formats used for comparison use polygon descriptions with room for six vertices (a total of 18 floats for coordinates of six points in three dimensions) even if the polygon has only three, four, or five vertices. All polygon descriptions are the same size in bytes in order to simplify memory allocation during terrain paging as discussed in Chapter IV. Excluding the counts that appear first in the file, the files contain polygon descriptions stored in quadtree heap-sort order with the polygon description at the 1000 meter resolution first, followed by the 500 meter resolution description for the same area, and so on for the 250 and 125 meter resolution descriptions.

C. TERRAIN DATA STRUCTURE

1. Description

The terrain data structure used in NPSNET is an array of quadtrees. The base array is 50 by 50 and corresponds to the area of the entire dataset; however, only a 16 by 16 subset of this array holds data at any one time. Changing the active area of the simulator is discussed in Chapter IV. Each element of the array contains a structure consisting of an array of 85 integers to hold the count of polygons at each quadtree node, and an array of 85 pointers that point to the first polygon belonging to each node. Each element of the array and its associated quadtree correspond to one square kilometer of terrain. This arrangement is depicted in Figure 6.7.

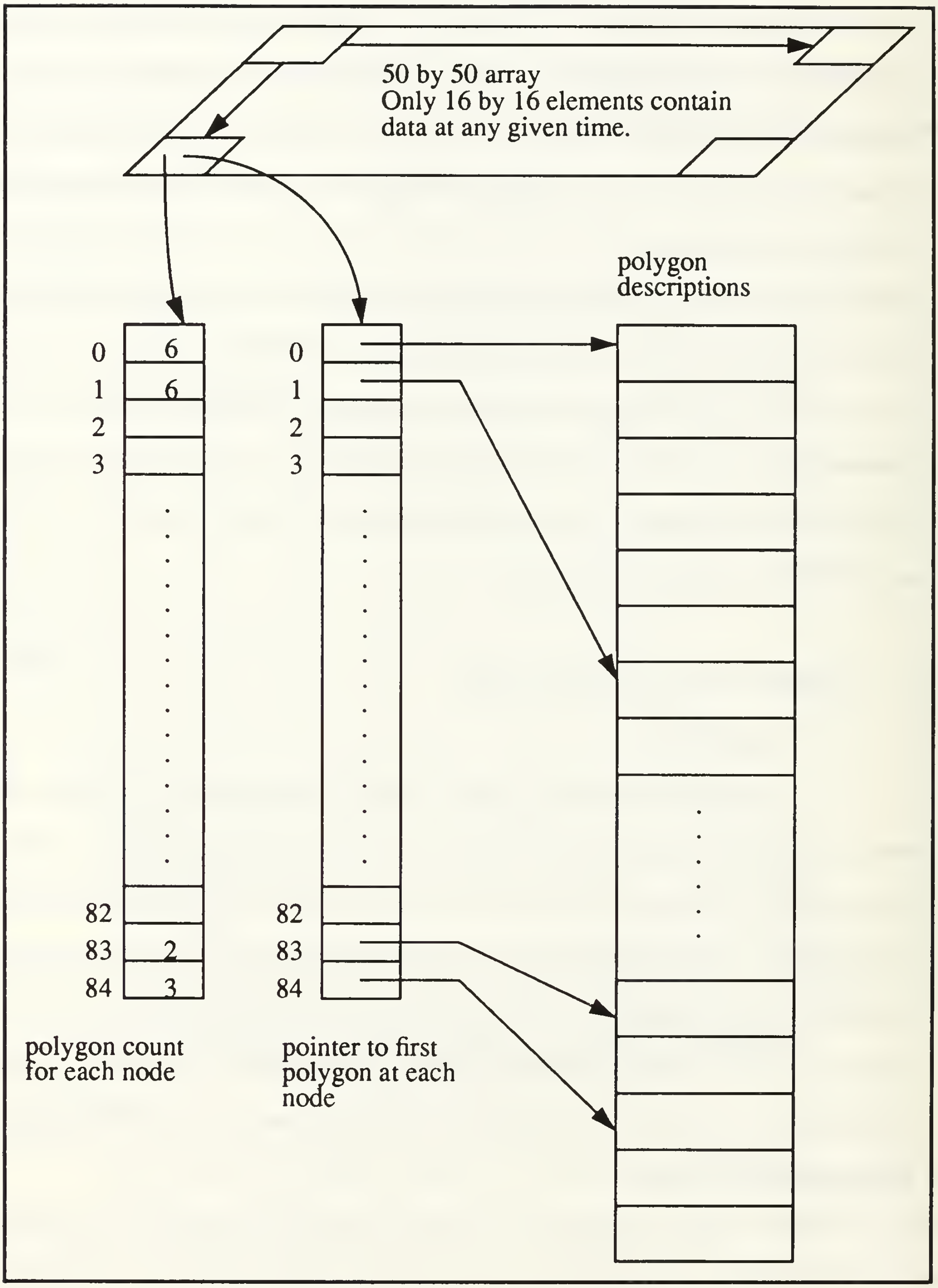


Figure 6.7. Terrain Data Structure in NPSNET

2. Accessing Data Files

The terrain data files are accessed and the data structure is filled in the following manner:

1. The 85 integers holding the polygon counts are block read from the data file into an array.

2. The total polygon count is read from the file.

3. Memory for that number of polygon descriptions is allocated and pointed to by the root node.

4. The polygon descriptions are block read from the file and pointed to by the root node.

5. The array containing the polygon counts is traversed and the corresponding pointers are set into the block of memory holding the polygon descriptions.

Initially 256 files are read in this manner to fill the active area of the data structure. During terrain paging, files corresponding to a strip of 16 square kilometers of terrain must be consecutively read in to the data structure in order to update the active area of the simulator.

VII. TERRAIN RENDERING IN NPSNET

Terrain rendering in NPSNET involves several steps. First, only the terrain that is actually in the field of view is rendered in order to reduce the number of polygons in each frame. Next the resolution of various parts of the terrain is determined. A determination is also made as to which fill polygons are needed to close gaps at resolution boundaries. Finally the terrain is actually rendered.

The examples shown in this chapter assume a 55 degree field of view and 125 meter resolution terrain rendered out to 1500 meters, 250 meter resolution terrain rendered from 1500 to 3000 meters, 500 meter resolution terrain rendered from 3000 to 4500 meters, and 1000 meter terrain rendered from 4500 to 6000 meters. The maximum range of terrain can not exceed beyond the edge of the terrain in the active area. All resolution levels do not have to be used and the ranges of the resolution levels can be varied; however, the algorithms used assume no 1000 meter square spans more than two resolution areas. This would affect the determination of which fill polygons to render.

A. TERRAIN POLYGON CULLING

Culling of terrain polygons outside of the field of view occurs in several steps in NPSNET. First a bounding box is established around the field of view as shown in Figure 7.1. Only terrain within this bounding box is potentially visible. This is a “rough cut” at polygon culling and may be all that is necessary in some applications (Jurewicz, 1990, pp. 13-16), but further refinement of the area actually visible is accomplished in NPSNET.

Further refinement involves checking each of the 1000 by 1000 meter squares contained within the bounding box. Each of these squares is checked to see if any of its

corners are contained within the field of view. This is done by calling a procedure that checks for the intersection of a point and a polygon (Fichten and Jennings, 1988 p. 95). This procedure is called using each corner of the 1000 meter squares in the bounding box as the point and the triangle composing the field of view as the polygon. If any of the corners of a 1000 meter square are within the field of view, then some part of the square is visible and the square is further processed for resolution and rendering.

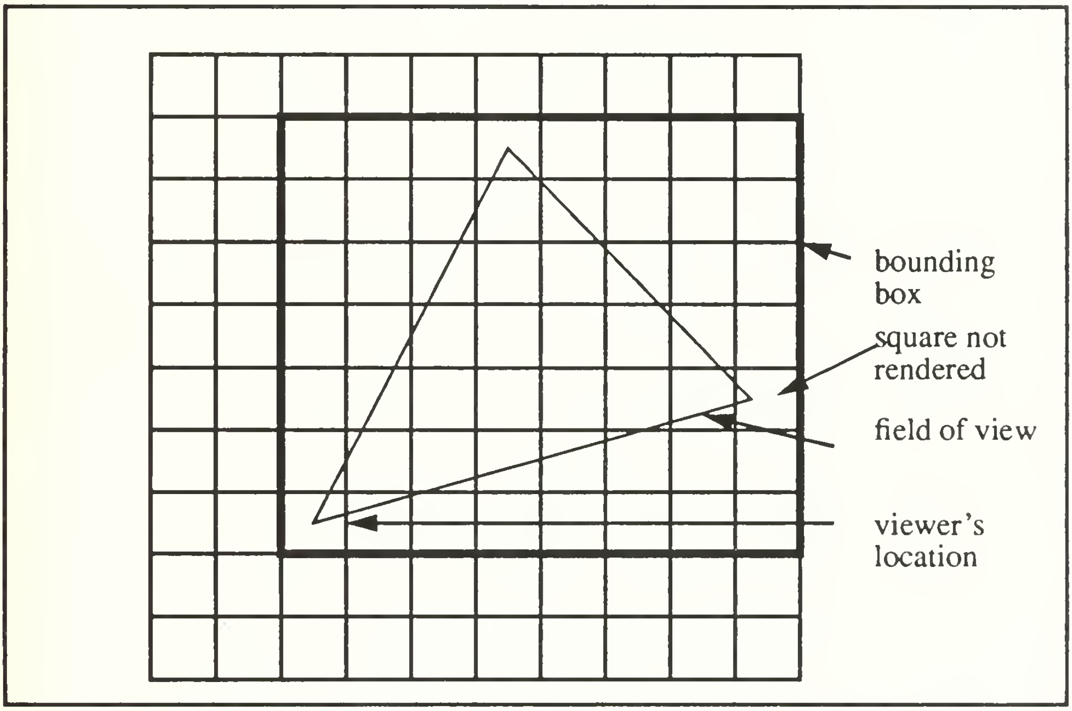


Figure 7.1. Bounding Box Surrounding Field of View

At certain directions of view the algorithm used for point-polygon intersection is not able to detect some squares near the viewer's location that are in the field of view. One of these cases is shown in Figure 7.2. A check is made for these special cases, and in these cases the appropriate 1000 meter squares are processed for resolution and rendering without further checking. In fact, the 1000 meter square containing the viewer's location is always further processed for resolution and rendering. This same problem occurs on the far

end of the field of view as shown in Figure 7.1, but these grid squares are ignored because of their distance from the viewpoint.

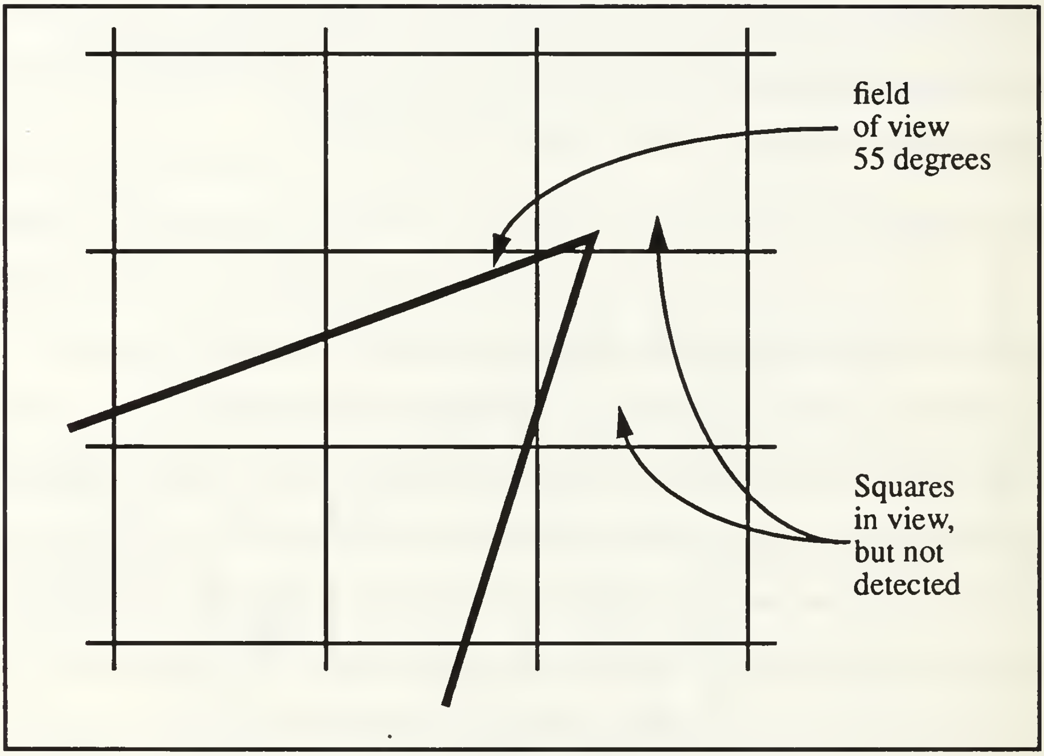


Figure 7.2. 1000 Meter Squares Not Detected by Point-Polygon Intersection

B. DETERMINING TERRAIN RESOLUTION

The next step is to determine the resolution of the terrain within each 1000 meter square. A 1000 meter square may eventually have parts of its terrain rendered at two different resolutions. The determination of resolution, essentially determining which nodes of the quadtree to render, is performed using an algorithm that checks for intersection of the quadtree nodes with concentric circles surrounding the viewer's location as shown in Figure 7.3. The algorithm simply checks the intersection of a circle and a rectangle

(Shaffer, 1990, pp. 51-53). The concentric circles correspond to the ranges of the various resolutions.

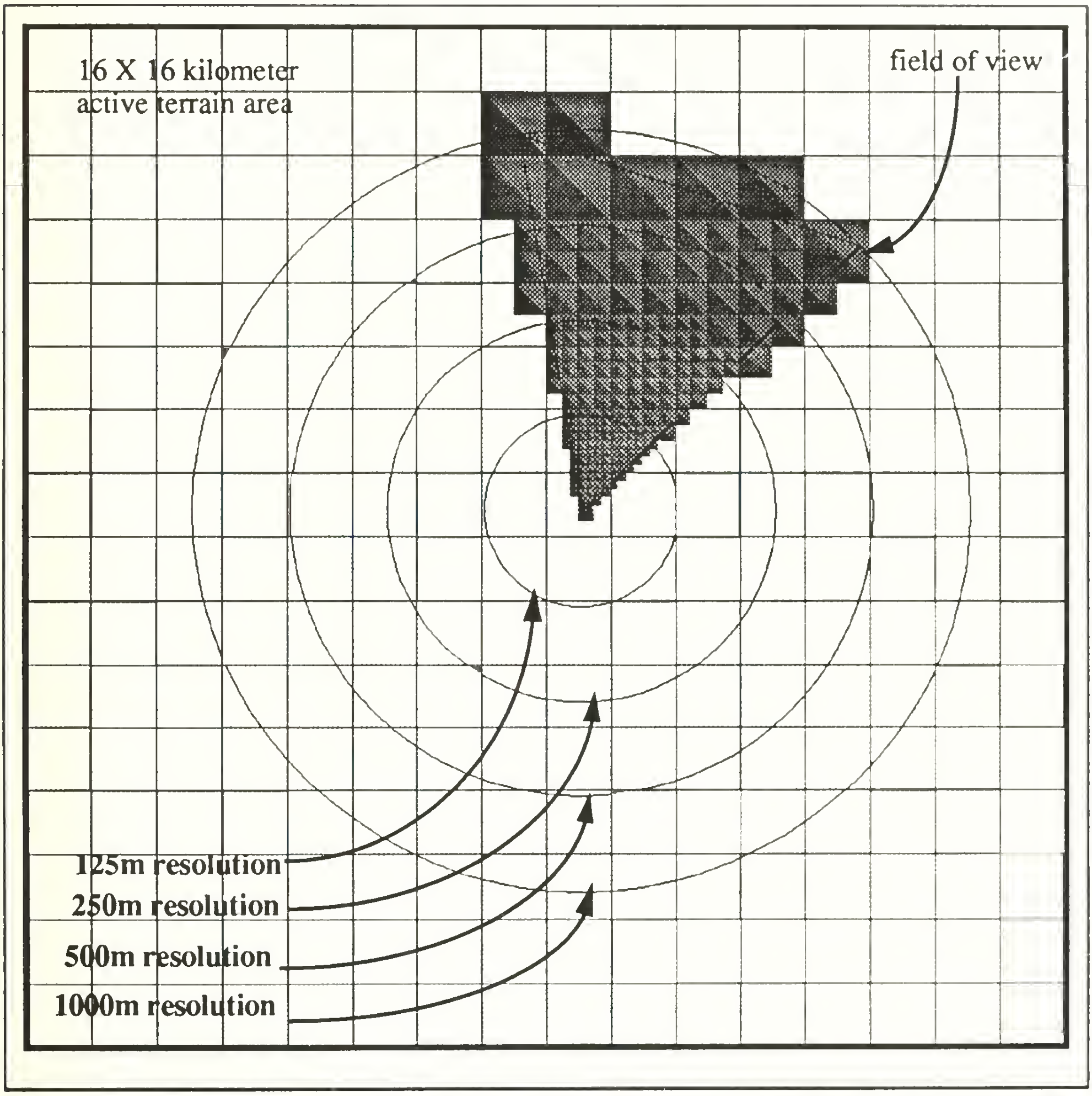


Figure 7.3. Conceptual Two-Dimensional View of Terrain Active Area, Field of View, and Terrain Rendered at Different Resolutions

The circle-rectangle intersection algorithm and the point-polygon intersection algorithm are applied repetitively in order to render only terrain within the field of view and

to render that terrain at the appropriate resolution levels. A two dimensional view of the active area of the terrain, the field of view, and the terrain rendered is shown in Figure 7.3. Pseudocode displaying the manner in which the algorithms are applied is shown in Figures 7.4 and 7.5.

```
for all 1000m squares in the active area
    if viewer's location is in this square then
        process square for resolution and rendering
    else
        if this is a special case near viewer's location then
            process square for resolution and rendering
    else
        for all corners of this square
            if corner is inside field of view then
                process square for resolution and rendering
```

Figure 7.4. Algorithm to Determine Which 1000 Meter Squares Are Within the Field of View

The cost of determining what terrain is within the field of view and should be rendered must outweigh the cost of rendering terrain that is actually not in the field of view. If "fine tuning" which terrain to render is too expensive, it is preferred to perform a simpler determination of what to render. NPSNET proved to be a graphics bound versus computation bound program early in its development, so every attempt is made to reduce calls to graphics procedures. Limiting the amount of terrain rendered to the absolute minimum is one of these attempts.

```
if 1000m node is inside 1000m resolution circle then
    if 1000m node is inside 500m resolution circle then
        if 1000m node is inside 250m resolution circle then
            if 1000m node is inside 125m resolution circle then
                for each 250m node
                    if 250m node is inside 125m resolution circle then
                        if 250m node contains view location
                            render its four 125m nodes
                        else
                            for each 125m node
                                if 125m node is in field of view
                                    render 125m node
                        if all corners of 250m node not within 125m
                                             resolution circle then
                            render appropriate fill polygons
                    else /* 250m node is not in 125 resolution circle */
                        render 250m node
            else /* 1000m node is within 250m resolution circle */
                for each 500m node
                    if 500m node is within 250m resolution circle then
                        for each 250m node
                            if 250m node is in field of view then
                                render 250m node
                        if all corners of 500m node not within 250m
                                             resolution circle then
                            render appropriate fill polygons
                    else /* 500m node is not within 250m resolution circle */
                        render 500m node
        else /* 1000m node is inside 500m resolution circle */
            for each 500m node
                if 500m node is in field of view then
                    render 500m node
            if all corners of 1000m node not within 500m resolution circle then
                render appropriate fill polygons
    else /* node is within 1000m resolution */
        render 1000m node
```

Figure 7.5. Algorithm for Determining Which Quadtree Nodes Are Rendered

C. DETERMINING REQUIRED FILL POLYGONS

The algorithm used for circle-rectangle intersection in a modified version is used to determine which fill polygons, if any, need to be rendered in order to close gaps at the boundaries of terrain rendered at different resolutions. The modified version of the algorithm is applied after the original version has already determined if in fact a particular terrain square intersects a particular resolution circle. The algorithm is maintained in two separate versions because the modified version, which determines which specific fill polygons are required, is more computationally expensive than the initial version. The initial version only checks if part of a terrain square lies within a given resolution circle. The required fill polygons correspond to which vertices of a square of terrain lie outside of a resolution circle. This is highlighted in Figures 7.6 and 7.7. A four element boolean array is used to record which fill polygons need to be rendered and which ones do not. Note that this algorithm is applied under the assumption that a square of terrain does not span more than two different resolutions; the algorithm becomes more complicated (and more computationally expensive) if this is not the case. In this version of NPSNET, where the largest square of terrain being checked is 1000 by 1000 meters, this means resolution circles must be at least 1414.21 meters apart (the length of the diagonal of a 1000 by 1000 meter square).

```
if part of this square is inside this resolution circle then

    apply modified circle-rectangle intersection algorithm

    if one or no corners are outside of this resolution circle then
        /* cases 1 and 2 from Figure 7.7 */
        no fill polygons are needed

    if two corners are outside of this resolution circle then
        /* case3 from Figure 7.7 */
        fill polygon between these two corners is needed

    if three corners are outside this resolution circle then
        /* case 4 from Figure 7.7 */
        two fill polygons between these three corners is needed

    if four corners are outside this resolution circle
    /* case 5 from Figure 7.7 */
    /* circle intersects a side of the terrain square, but not any corners */
        three fill polygons corresponding to three sides of square outside of
                        resolution circle are required
```

Figure 7.6. Determining Required Fill Polygons

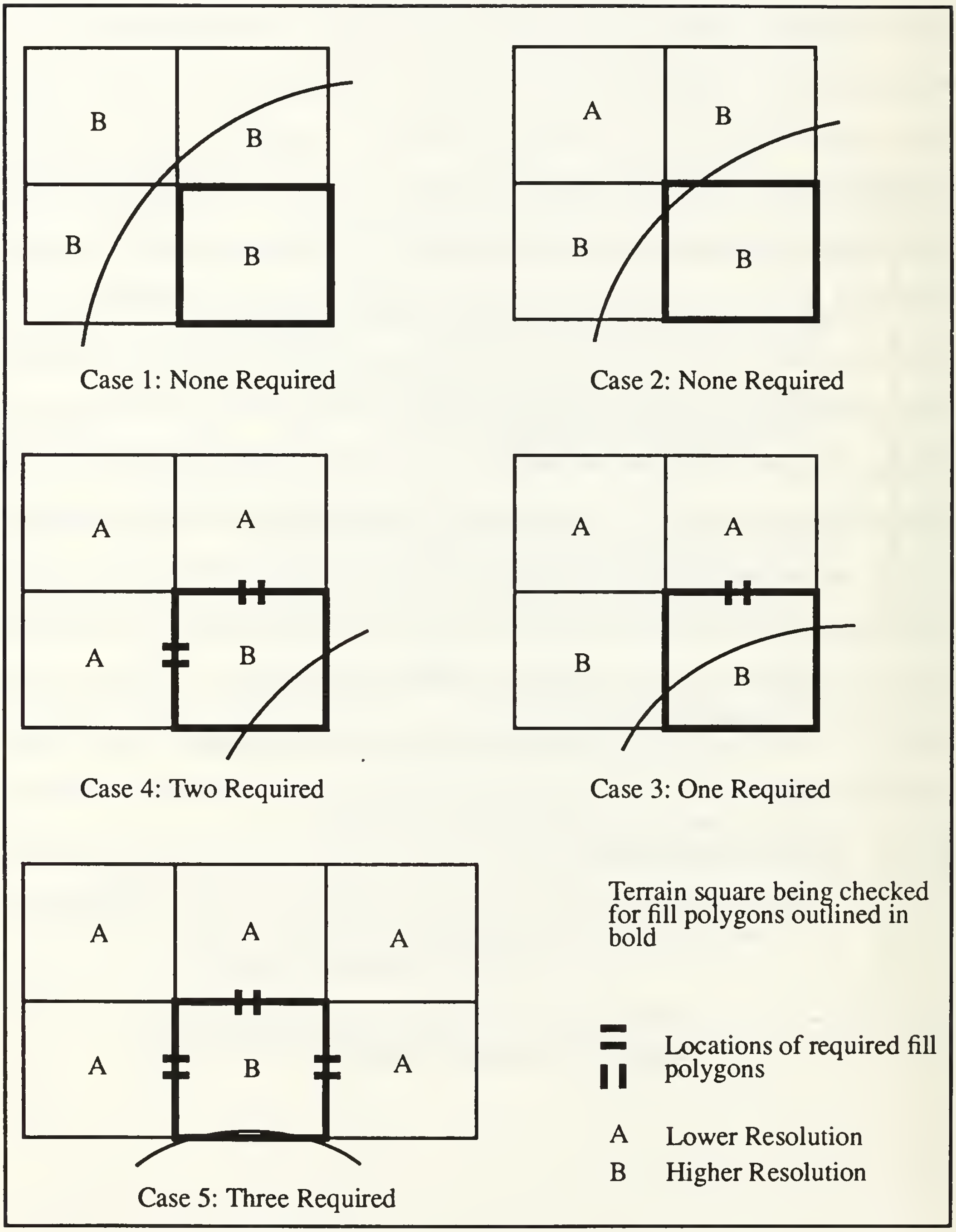


Figure 7.7. Circle-Rectangle Intersection and Required Fill Polygons

D. RENDERING TERRAIN SURFACE AND ROADS

Terrain in NPSNET is rendered using polygon drawing primitives. This involves passing the polygon normal vector and the vertex coordinates to the graphics hardware for transformation, lighting, and clipping. Special steps must be taken when rendering roads on terrain. In the dataset used in NPSNET, the polygons forming the road surface are coplanar with the polygons forming the underlying terrain surface. The roads must be "decaled" on the terrain. Decaling involves the following steps:

1. Drawing the underlying terrain polygons in the z-buffer modifying only color values and not range values.
2. Drawing the road polygons in the z-buffer modifying only color values and not range values.
3. Drawing the underlying terrain polygons in the z-buffer again modifying only range values and not color values.

These steps ensure that "polygon tearing" does not occur as a result of rendering coplanar polygons. Decaling involves rendering terrain twice and is therefore expensive to perform. For this reason roads in NPSNET are only rendered on terrain at the highest resolution, and not on terrain at the 250, 500, and 1000 meter resolution levels.

E. RENDERING OBJECTS ON MULTI-RESOLUTION TERRAIN

One problem quickly becomes apparent when rendering objects on multi-resolution terrain--the elevation of a given point can be different at different resolution levels. If the data for the elevation of objects is only available for one resolution level, then using this data to place objects on other resolution levels of terrain will place objects below the terrain surface or suspended in air. This problem is illustrated in Figure 7.8.

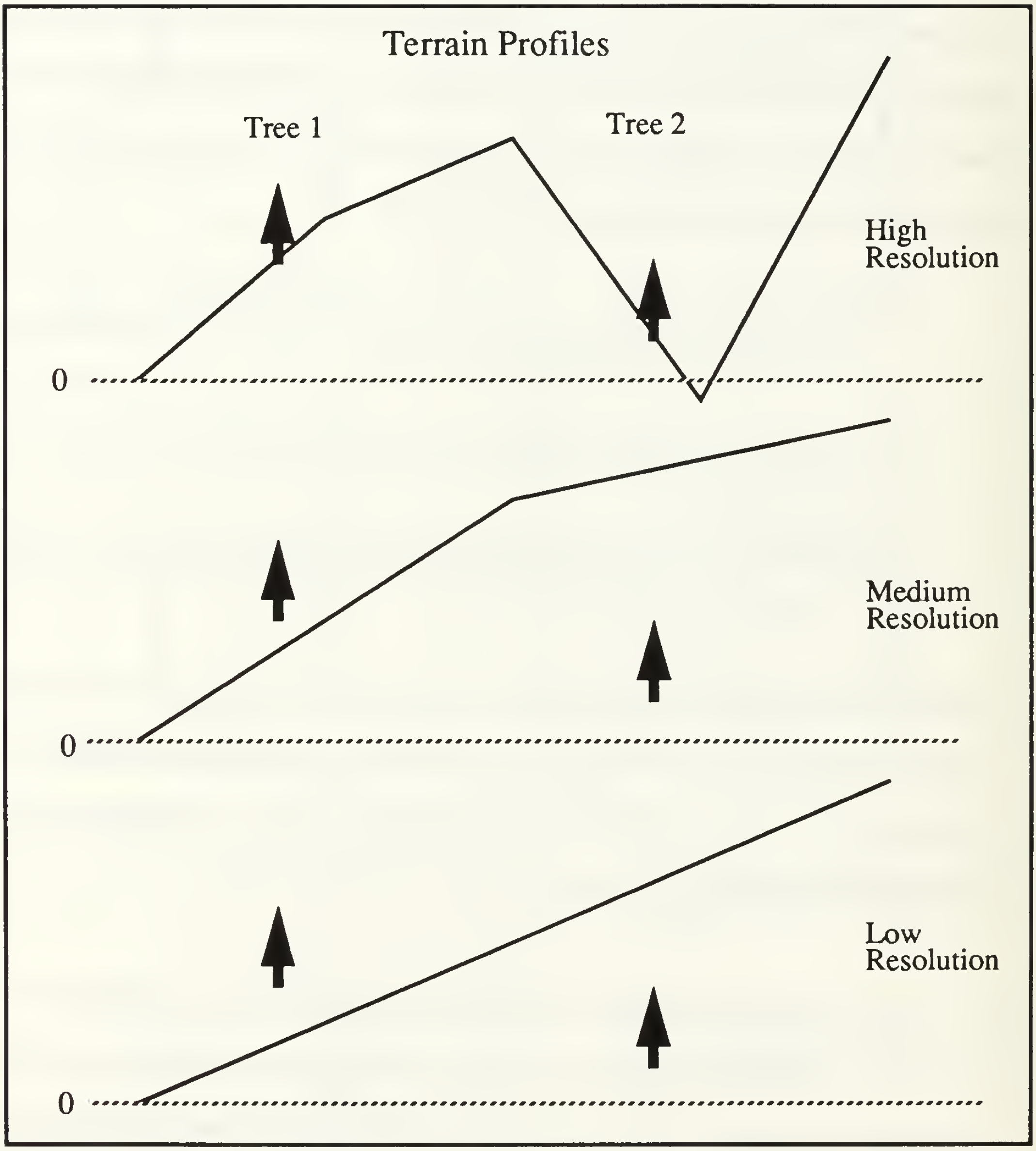
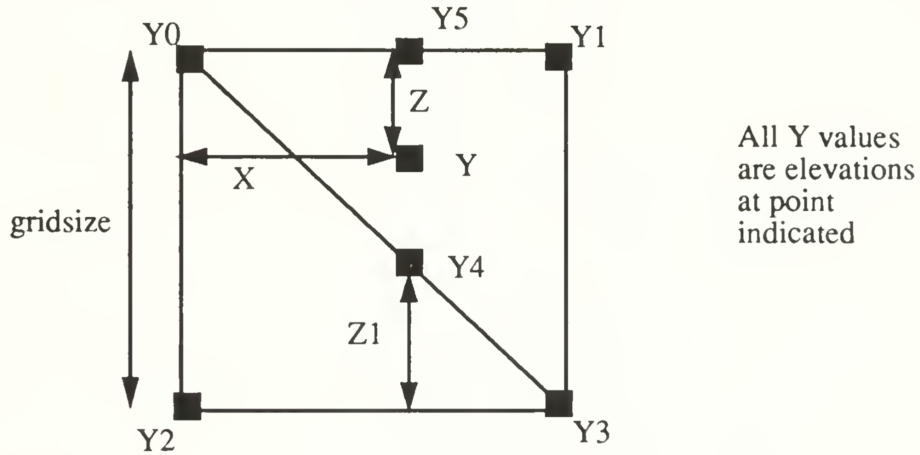


Figure 7.8. Placing Objects on Multi-Resolution Terrain Using Only High Resolution Elevations

This problem is solved in NPSNET by preprocessing the data regarding the location and elevation of objects. Four different elevations are generated for each instantiation of each object--one for each resolution level. The algorithm for determining these elevations is adapted from one used in MPS (Fichten and Jennings, 1988, p. 95) to determine the elevation of vehicles as they move over terrain. The algorithm is shown in Figure 7.9. The algorithm is applied for each level of resolution. The generated elevations are stored in a file along with the coordinate location of each object.




```
Z1 = gridsize - X
percentage = X / gridsize
if x = 0.0 then
    Y = Y0 + ((Y2 - Y0) * Z) / gridsize)
else
    Y4 = Y0 + ((Y3 - Y0) * percentage)
    if (Z < X)
        Y5 = Y0 + ((Y1 - Y0) * percentage)
        Y = Y5 + ((Y4 - Y5) * (Z / X))
    else
        Y5 = Y2 + (Y3 - Y2) * percentage
        Y = Y5 + (((Y4 - Y5) * (gridsize - Z)) / Z1)
```

Figure 7.9. Algorithm for Determining Elevation Within Square of Terrain

VIII. SUMMARY

A. RESULTS

1. Terrain Paging

Terrain paging allows NPSNET to use the entire available dataset for simulation. Dividing the terrain data into files based on one square kilometer areas of terrain simplifies implementation and editing the data since this follows the convention of the military grid reference system. On a multiple processor machine, terrain paging has a negligible effect on simulator performance. Four parallel processes, one for paging terrain in each of the four cardinal directions, allow motion to continue while terrain is paged in the direction of travel. In this implementation, files used for terrain data contain one square kilometer of terrain data, in four resolutions, stored in quadtree heap-sort order. The algorithms used for terrain paging will support other data configurations since paging only involves accessing files and is not dependent on the underlying structure of the data.

2. Multiple Resolution

The implementation of a hierarchical data structure based on quadtrees provides a means to implement multiple resolutions of data. The availability of multiple resolution provides a means to reduce the number of polygons composing each individual frame. Figure 8.1 shows a comparison of the number of polygons forming the terrain surface under different resolution configurations. The numbers shown represent only terrain surface polygons within the field of view. This includes fill polygons in configurations with more than one resolution. These numbers do not include polygons for road surfaces or any objects placed on the terrain.

Resolution Configuration	Distance	Number of Polygons
High Resolution	0-2500 meters	415
High Resolution	0-6000 meters	2080
High Resolution Med High Resolution	0-3000 meters 3000-6000 meters	1008
High Resolution Med High Resolution Med Low Resolution	0-2000 meters 2000-4000 meters 4000-6000	641
High Resolution Med High Resolution Med Low Resolution Low Resolution	0-1500 meters 1500-3000 meters 3000-4500 meters 4500-6000 meters	435

Figure 8.1. Comparison of Number of Polygons in Terrain Surface

The number of polygons in the terrain surface affect the frame rate of NPSNET. Figure 8.2 shows a comparison of frame rates for different resolution configurations. The frame rates are measured from the same view point on the terrain and viewing in the same direction. Several random vehicles are also present in the frames. No other objects are rendered in the test scenes. The measurements were taken on an IRIS 4D/ 240VGX.

The results show that terrain in four resolutions out to 6000 meters, can be rendered at approximately the same expense as terrain in one resolution rendered out to

2500 meters. This validates the incorporation of multiple resolution to increase the range of views; however, placing objects and vehicles on the additional terrain available will reduce the frame rate. Any other computations which vary depending on the amount of terrain in view will also affect the frame rate.

Resolution Configuration	Distance	Frames per Second
High Resolution	0-2500 meters	6.3
High Resolution	0-1000 meters	8.2
Med High Resolution	1000-2500 meters	
High Resolution	0-3000 meters	4.4
Med High Resolution	3000-6000 meters	
High Resolution	0-2000 meters	5.4
Med High Resolution	2000-4000 meters	
Med Low Resolution	4000-6000	
High Resolution	0-1500 meters	6.5
Med High Resolution	1500-3000 meters	
Med Low Resolution	3000-4500 meters	
Low Resolution	4500-6000 meters	

Figure 8.2. Comparison of Frame Rates

Rendering a large area of terrain and placing features and vehicles on all terrain in the field of view adversely affects the frame rate of the simulator if the density of these objects is large. Ways to overcome this include not rendering features and vehicles on lower resolution terrain, or rendering lower resolution versions of these objects.

B. CONCLUSIONS

The implementation of terrain paging in NPSNET efficiently allows the use of a large area of terrain for simulation. With this system of terrain paging, the size of the terrain area used is only limited by the amount of secondary memory. Using parallel processing for updating the active area of terrain prevents waiting for additional terrain to become available for rendering.

Quadtrees provide a means to effectively implement multiple resolutions of terrain. Quadtrees are particularly suited for terrain data applications since they can be used to divide the terrain into a system of squares in the same manner as grid reference systems. Many algorithms and techniques which support quadtrees have been developed and these assist in efficient implementations.

The determination of resolution areas can be computationally expensive. The savings from rendering terrain at different resolutions must outweigh the cost of determining resolution areas. A quick, rough determination may be better than a stringent one. The implementation of multiple resolutions must support the objectives of the simulator. The way in which multiple resolutions of data is implemented should be derived from these goals.

C. SUGGESTIONS FOR FUTURE WORK

This implementation of terrain paging and hierarchical data structures in NPSNET can serve as the basis for further research in these and associated areas. Further work should involve improving efficiency, reducing memory requirements, and adding additional capabilities.

Full quadtrees are used to store terrain data in this implementation even though a low resolution description of some areas of terrain would suffice. An example is large areas of water; these areas could be described in only the lowest resolution and no surface detail

would be lost. Such terrain descriptions could be developed by applying adaptive refinement to the original terrain data. Using quadtrees that are not full would change many of the algorithms used in this implementation, but less memory would be needed to store the terrain descriptions.

The ability to display multiple resolution terrain is important when using display devices with wider fields of view. Such devices include head worn goggles and multiple screen displays. When the field of view is wide, not only can lower resolution terrain be displayed in areas far away from the viewer, but also along the edges of the field of view. This would result in a configuration where the highest resolution terrain is displayed near and directly in front of the viewer. Lower resolution terrain would be displayed at greater distances and to the sides of the viewer.

The determination of what terrain to render and the resolution of that terrain requires extensive computation. This implementation uses a circle-rectangle intersection algorithm and a point-polygon intersection algorithm extensively. Any improvement in these algorithms would have a positive effect on the performance of NPSNET.

The terrain paging algorithm could be adapted to support a more realistic aircraft view. Adding additional, lower resolution terrain descriptions--two by two, four by four, eight by eight kilometers, etc.--would enable the simulator to provide longer range views at higher altitudes. The goal should be to allow views out to 26 nautical miles, the generally accepted limit of view from an aircraft (Schachter, 1983, p. 75). The incorporation of such an air view would also require the view volume to vary based on the altitude of the aircraft and the pilot's angle of view toward the earth.

The attributes which describe terrain polygons can be used to determine mobility across terrain. A system could be implemented which uses these attributes to increase or decrease mobility of vehicles as they move over terrain.

APPENDIX A

TERRAIN POLYGON ATTRIBUTES

ATTRIBUTE	TYPE	DESCRIPTION
X INDEX	Integer	Index of upper left corner of square containing polygon in 125's of meters
Z INDEX	Integer	Index of upper left corner of square containing polygon in 125's of meters
NUMBER OF POINTS	Integer	Number of vertices of the polygon
COLOR	Integer	Color of the polygon 6 - Paved Roads 12 - Sand 14 - Water 17 - Dirt Roads
LEVEL	Integer	Relative level of the polygon 1 - earth or water 2 - fill polygons 10 - roads

ATTRIBUTE	TYPE	DESCRIPTION
TRIANGLE	Character	Which triangle of the square this polygon belongs to u - upper l - lower f - fill polygon
NORMAL	3 floats	X,Y, and Z values of polygon normal vector
VERTICE	3 Floats	X, Y, Z coordinates of each vertex up to a maximum of six

APPENDIX B

TERRAIN DATAFILE FORMAT

Terrain surface polygons are described using the following structure. The allowable values for the fields are discussed in Appendix B.

POLYGON DESCRIPTION

FIELD	NUMBER	TYPE	DESCRIPTION
xgrid	1	4 byte integer	X index of upper left corner of square containing polygon in 125's of meters.
zgrid	1	4 byte integer	Z index of upper left corner of square containing polygon in 125's of meters.
num_of_pnts	1	4 byte integer	Number of points in this polygon
color	1	4 byte integer	Color of polygon
level	1	4 byte integer	Terrain level of this polygon (not resolution level).
which_tri	1	4 byte character	Which triangle of the square contains this polygon.

FIELD	NUMBER	TYPE	DESCRIPTION
normal	1	3 X 4 bytes float	Unit polygon normal vector
points	6	3 X 4 byte float	Coordinates of polygon vertices in three dimensions

Six vertices are used because this is the maximim number of sides a road decal can have. Overlaying a rectangle on a triangle produces a polygon of intersection with from three to six sides. If a polygon has less than six vertices, the coordinate values for the unused vertices are 0.0, 0.0, 0.0.

The individual data files used in NPSNET contain surface data for one square kilometer of terrain. The format for the binary files is as follows:

FIELD	NUMBER	TYPE	DESCRIPTION
nodecount	85	4 byte integer	Count of polygons in each node of four level quadtree.
totalcount	1	4 byte integer	Total polygon descriptions in file.
polygon	totalcount	108 byte polygon description as indicated above	Multi-resolution description of terrain in this square kilometer.

The polygons are stored in the files in quadtree heap-sort order. The first polygon descriptions in the files are descriptions for the terrain at the 1000 meter resolution level, followed by the 500 meter resolution level, 250 meter resolution level, and 125 meter resolution level.

The first four polygon descriptions in each node at the 1000, 500, and 250 meter resolution levels are for fill polygons. These nodes each contain a total of six polygons--four fill polygons and two terrain surface polygons. The nodes at the 125 meter level do not contain fill polygons, but may contain polygons which describe roads. An annotated extract from a text version of a data file begins on the next page.

```
.
.
.
2
2
2
2
6
2
2
2
2
7
6
2
2
2
6
5
2
4
276
208 80 3 12 2 f
0.000000 0.000000 1.000000
26500.000000 585.216003 10000.000000
26000.000000 657.148804 10000.000000
27000.000000 638.251221 10000.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
.
.
.
.
208 80 3 12 1 1
0.018684 0.988681 0.148867
27000.000000 638.251221 10000.000000
26000.000000 657.148804 10000.000000
27000.000000 487.679993 11000.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
```

Count of polygons in each quadtree node

Total polygon descriptions in file

Fill polygon for 1000 meter resolution

Polygon description for lower triangle in this 1000 X 1000 meter square

```
208 80 3 12 1 u
0.029873 0.989996 0.137900
26000.000000 517.855225 11000.000000
26000.000000 657.148804 10000.000000
27000.000000 487.679993 11000.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
```

Polygon description for upper triangle in this 1000 X 1000 meter square.

```
.

.
210 80 3 12 2 f
0.000000 0.000000 1.000000
26375.000000 606.552002 10000.000000
26250.000000 623.011230 10000.000000
26500.000000 585.216003 10000.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
210 80 3 12 2 f
1.000000 0.000000 0.000000
26500.000000 540.410400 10125.000000
26500.000000 585.216003 10000.000000
26500.000000 562.965637 10250.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
210 80 3 12 2 f
0.000000 0.000000 -1.000000
26375.000000 524.560791 10250.000000
26500.000000 562.965637 10250.000000
26250.000000 585.216003 10250.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
210 80 4 12 2 f
-1.000000 0.000000 0.000000
26250.000000 573.023987 10125.000000
26250.000000 585.216003 10250.000000
26250.000000 623.011230 10000.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
```

Polygon descriptions for an entire 250 X 250 meter area.
Note first four polygons are fill polygons to mate terrain of different resolutions

```
210 80 3 12 1 1
0.148907 0.984958 0.087663
26500.000000 585.216003 10000.000000
26250.000000 623.011230 10000.000000
26500.000000 562.965637 10250.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
210 80 3 12 1 u
0.087663 0.984958 0.148907
26250.000000 585.216003 10250.000000
26250.000000 623.011230 10000.000000
26500.000000 562.965637 10250.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
```

Polygon description for an entire 250 X 250 meter area

```
.
.
208 87 3 12 1 1
-0.352554 0.920919 -0.166172
26000.000000 492.252014 10875.000000
26125.000000 562.660828 11000.000000
26125.000000 540.105591 10875.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
208 87 3 12 1 u
-0.331321 0.924328 -0.189326
26000.000000 492.252014 10875.000000
26000.000000 517.855225 11000.000000
26125.000000 562.660828 11000.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
208 87 3 17 10 1
-0.353757 0.920358 -0.166724
26000.000000 492.250000 10875.000000
26000.990234 492.809998 10875.990234
26002.029297 493.029999 10875.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
```

Polygon descriptions for an entire 125 X 125 meter area. Note two polygons describing underlying terrain and two polygons descibing a road surface.

```
208 87 3 17 10 u
-0.332016 0.923850 -0.190439
26000.000000 492.250000 10875.000000
26000.000000 492.649994 10876.940430
26000.990234 492.809998 10875.990234
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
0.000000 0.000000 0.000000
```

LIST OF REFERENCES

Airey, John M., *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations*, Ph.D. Dissertation, University of North Carolina, Chapel Hill, North Carolina, 1990.

Air Force Wright Aeronautical Laboratories Report TR-86-1177, *Hierarchical Data Structures for a Digital Terrain Map System*, by Tiburon Systems Inc., December 1986.

Cheeseman, Curtis P., *Moving Platform Simulator III: An Enhanced High-Performance Real-Time Simulator with Multiple Resolution Display and Lighting*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1990.

DeHaemer, Michael J., Jr., *Simplification of Objects Rendered by Polygonal Approximation*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1989.

Defense Mapping Agency, Specification PS/ICD/200 PS/ICF/200, *Product Specifications for Digital Terrain Elevation Data (DTED)*, April 1986.

Fichten, Mark A. and David H. Jennings, *Meaningful Real-Time Graphics Workstation Performance Measurements*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1988.

Fuchs, Henry, Kedem, Zvi M., and Naylor, Bruce F., *On Visible Surface Generation by A Priori Tree Structures,* Proceedings of SIGGRAPH, Volume 14, Number 3, July 1980, pp.124-133.

Gargantini, Irene, *An Effective Way to Represent Quadtrees*, Communications of the ACM, Volume 25, Number 12, December 1982, pp. 905-910.

Jurewicz, Thomas A., *A Real-Time Autonomous Underwater Vehicle Dynamic Simulator*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1990.

King, David M. and Prevatt, Richard M. III, *Rapid Production of Graphical User Interfaces*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1990.

Lang, Eric and Wever, Peter, *SDIS Version 3.0 User's Guide*, BBN Systems and Technologies, Bellevue, Washington, August 1990.

Oliver, Michael R. and Stahl, David J., *Interactive, Networked, Moving Platform Simulators*, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1987.

Samet, Hanan, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, Massachusetts, 1990a.

Samet, Hanan, *Applications of Spatial Data Structures*, Addison-Wesley, Reading, Massachusetts, 1990b.

Schachter, Bruce J., *Computer Image Generation,* John Wiley & Sons, New York, 1983.

Shaffer, Clifford, "Fast Circle-Rectangle Intersection", *Graphics Gems,* Ed. Andrew Glassner, Academic Press, Boston, 1990, pp. 51-53.

Smith, Douglas B., and Dale G. Streyle, *An Inexpensive Real-Time Interactive Three-Dimensional Flight Simulation System*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1987.

Thorpe, Jack A., "The New Technology of Large Scale Simulator Simulator Networking: Implications for Mastering the Art of Warfighting," *Proceedings of the Ninth Interservice Industry Training Systems Conference*, November 1987.

Weeks, Gordon K., Jr. and Charles E. Phillips, Jr., *The Command and Control Workstation of the Future: Subsurface and Periscope Views*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1989.

Winn, Michael C. and Strong, Randolph P., *Moving Platform Simulator II: A Networked Real-Time Simulator with Intervisibility Displays*, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1989.

Zyda, Michael J., Graphics Course Notes, Book 7, Naval Postgraduate School, Monterey, California, 1991a.

Zyda, Michael J., Graphics Course Notes, Book 9, Naval Postgraduate School, Monterey, California, 1991b.

Zyda, Michael J. and Pratt, David R., *NPSNET: A 3D Visual Simulator for Virtual World Exploration and Experimentation*, 1991 Society for Information Display International Symposium Digest of Technical Papers, Volume XXII, May 1991, pp 361-364.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Dudley Knox Library Code 52 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Dr. Michael J. Zyda Code CS/Zk, Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	4
4.	David R. Pratt Code CS/Pr, Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	5
5.	Captain Randall L. Mackey Rt. 2, Box 25 Bancroft, Nebraska 68004	2